Trustworthy Machine Learning

Charles Meyers

Doctoral Thesis, June 2025
Department of Computing Science
Umeå University
Sweden

Department of Computing Science
Umeå University
SE-901 87 Umeå, Sweden

cmeyers@cs.umu.se

ISBN (print) 978-91-8070-722-0
ISBN (digital) 978-91-8070-723-7
ISSN 0348-0542
UMINF 25.10

Printed by Scandinavian Print Group, 2025

Abstract

This thesis studies adversarial robustness, privacy, and reproducibility in safety-critical machine learning systems, with particular emphasis on computer vision, anomaly detection, and evasion attacks through a series of papers. The work begins by analysing the practical costs and benefits of defence strategies against adversarial attacks, revealing that common robustness metrics are poor indicators of real-world adversarial performance (Paper I). Through large-scale experiments, it further demonstrates that adversarial examples can often be generated in linear time, granting attackers a computational advantage over defenders (Paper II). To address this, a novel metric—the Training Rate and Survival Heuristic (TRASH)—was developed to predict model failure under attack and facilitate early rejection of vulnerable architectures (Paper III). This metric was then extended to real-world cost, showing that adversarial robustness can be improved using low-cost, low-precision hardware without sacrificing accuracy (Paper IV).

Beyond robustness, the thesis tackles privacy by designing a lightweight, client-side spam detection model that preserves user data and resists several classes of attacks without requiring server-side computation (Paper V). Recognizing the need for reproducible and auditable experiments in safety-critical contexts, the thesis also presents `deckard`, a declarative software framework for distributed and robust machine learning experimentation (Paper VI).

Together, these contributions offer empirical techniques for evaluating and improving model robustness, propose a privacy-preserving classification strategy, and deliver practical tooling for reproducible experimentation. Ultimately, this thesis advances the goal of building machine learning systems that are not only accurate, but also robust, reproducible, and trustworthy.

Sammanfattning

Denna avhandling studerar robusthet, integritet och reproducerbarhet i säkerhetskritisk maskininlärning, med särskild tonvikt på datorseende, avvikelsedetektering och undvikande attacker.

Arbetet inleds med att analysera de praktiska kostnaderna och fördelarna med försvarsstrategier mot attacker, vilket visar att vanliga mått på robusthet är dåliga indikatorer på verklig prestanda i attacker (Artikel I). Genom storskaliga experiment visar arbetet vidare att exempel på attacker ofta kan genereras i linjär tid, vilket ger angripare en beräkningsfördel gentemot försvarare (Artikel II). För att hantera detta presenterar avhandlingen ett nytt mått – Training Rate and Survival Heuristic (TRASH) – för att förutsäga modellfel under attack och underlätta tidigt avvisande av sårbara arkitekturer (Artikel III). Detta mått utvidgades sedan till verkliga kostnader, vilket visar att robusthet i attacker kan förbättras med hjälp av billig hårdvara med låg precision utan att offra noggrannheten (Artikel IV).

Utöver robusthet behandlar avhandlingen integritet genom att utforma en lättviktig klientbaserad modell för spamdetektering som bevarar användardata och står emot flera klasser av attacker utan att kräva att beräkningar görs på serversidan (Artikel V). Som svar på behovet av reproducerbara och granskningsbara experiment i säkerhetskritiska sammanhang presenterar avhandlingen även `deckard`, ett deklarativt programvaruramverk för distribuerade och robusta maskininlärningsexperiment (Artikel VI).

Tillsammans erbjuder dessa bidrag empiriska tekniker för att utvärdera och förbättra modellers robusthet, föreslår en integritetsbevarande klassificeringsstrategi och levererar praktiska verktyg för reproducerbara experiment. Sammantaget främjar avhandlingen målet att bygga maskininlärningssystem som inte bara är korrekta, utan också robusta, reproducerbara och pålitliga.

Preface

This thesis is based on the following papers:

Paper I *Safety-critical computer vision: an empirical survey of adversarial evasion attacks and defenses on computer vision systems.* Charles Meyers, Tommy Löfstedt, and Erik Elmroth. Artificial Intelligence Review, Volume 56, 2023.
Available online:
https://link.springer.com/content/pdf/10.1007/s10462-023-10521-4.pdf

Paper II *Massively parallel evasion attacks and the pitfalls of adversarial retraining.* Charles Meyers, Tommy Löfstedt, and Erik Elmroth. EAI Endorsed Transactions on Internet of Things, Volume 10, 2024.
Available online:
https://publications.eai.eu/index.php/IoT/article/view/6652

Paper III *Training Rate and Survival Heuristic for Inference and Robustness Evaluation (Trashfire).* Charles Meyers, Mohammad Reza Saleh Sedghpour, Tommy Löfstedt, and Erik Elmroth. IEEE International Conference on Machine Learning and Cybernetics (ICMLC), Volume 1, 2024.
Available online: https://ieeexplore.ieee.org/document/10935101
Open access preprint: https://arxiv.org/abs/2401.13751

Paper IV *A Cost-Aware Approach to Adversarial Robustness in Neural Networks.* Charles Meyers, Mohammad Reza Saleh Sedghpour, Erik Elmroth, and Tommy Löfstedt. Submitted for publication September 2024.
Open-access preprint: https://arxiv.org/abs/2409.07609

Paper V *A Tiny, Client-Side Classifier.* Charles Meyers, Aaron P. MacSween, Erik Elmroth, and Tommy Löfstedt. Manuscript, Umeå University, Sweden, 2024. *Manuscript*

Paper VI *Deckard: A tool for robust, declarative, and reproducible AI.* Charles Meyers. Manuscript, Umeå University, Sweden, 2025. *Manuscript*

Acknowledgements

The first thanks go to Tommy and Erik, my PhD advisors, for guiding me through this whole process and offering feedback that was occasionally encouraging, frequently humbling, and always technically correct. Endless thanks also to my colleagues in the Autonomous Distributed Systems Laboratory for the thoughts they inspired, the questions they raised, and the stubborn determination they provoked.

By no means did I get here alone, and I'd like to thank some educators who spent countless hours dragging me this far: Christie Calloway, who somehow managed to inject her awe of mathematics into middle school students; Jonathan Williams, who looked the other way while I spent Algebra class programming on a TI-83 instead of solving for x; Lenora Murray, who ran our high school math club, taught me statistics, and repeatedly entrusted a teenage me with vaguely-supervised travel; Dr. Samar ElHitti, who taught me linear algebra and convinced the university to give me unrestricted use of their supercomputer; Dr. Johann Thiel, who taught me optimisation and modelling and kept the college math club alive with free pizza; and Dr. Viviana Acquavia, for insisting that I finish what I started.

To some colleagues who deserve equal parts praise and apology: Dr. Abel Souza, for insisting I use a modern development environment and gifting me his lease; Dr. Mohammad Reza, for contributions regarding the mystical realm of "the cloud"; Dr. Aryaman Jal (and, let's be honest, Tommy too), for tolerating my mathematical notation long enough to figure out what I was trying to say; and Aaron MacSween, who kept me grounded by asking the important questions—usually variations of "why are you doing this again?"

A very special thanks goes to Samantha Wolff, Forrest Crellin, Felix Barbut, Domenique André, and Suga the Cat for their years of dedicated service in enabling my relaxation, socialisation, and, most importantly, procrastination.

Finally, I must thank my family – for the genetic material, for access to early childhood reading material, and for motivating questions like, "When are you going to find a 'real' job?" and 'Can you help me fix my internet?"'

Y'all made this possible. Well, at least, you made it bearable.

Thanks.

Funding Statement

Financial support has been provided in part by the Knut and Alice Wallenberg Foundation grant number 2019.0352 and by the eSSENCE Programme under the Swedish Government's Strategic Research Initiative.

Glossary

Sample A single data point or observation, typically consisting of one or more features and, in supervised learning, a label (think: rows in a spreadsheet).

Feature An individual measurable property or characteristic of the data used by a model to make predictions (think: columns in a spreadsheet).

Training set A collection of data used to teach a machine learning model how to make predictions.

Validation set A subset of the data used to tune hyper-parameters and monitor the model's performance during training without affecting the final evaluation.

Testing set A separate set of data used to evaluate how well the model performs on new, unseen inputs.

Label Either the predicted or ground truth for a given sample. For classification problems, this is a category like "cat" or "dog", often represented as a binary list with each class having one entry in said list. For regression problems, this is one or more numerical values.

Model Model: A machine learning model consists of two main parts: a function that maps inputs to outputs (called a hypothesis function), and a way to measure how well it performs (called a loss function). Together, these components allow the model to learn patterns from data and make predictions or decisions on new inputs.

Classifier A type of model that assigns input data to one of several predefined categories or classes.

Regressor A type of model that assigns one or more numeric output values to a set of input data.

Model Parameter An internal value adjusted during training that helps the model improve its predictions.

Hyper-parameter A predefined setting that controls how the model behaves during training or evaluations, such as the learning rate or the number of layers.

Optimisation The process of adjusting the model's internal parameters in order to minimize error and improve performance based on a chosen loss function.

Training The process of teaching a machine learning model by providing it with data (a training set) and adjusting its parameters through optimization to minimize errors in predictions.

Accuracy The fraction of correct predictions made by the model out of all predictions.

Failure Rate The number of expected number of failures over time.

Time to Failure The duration a system operates before a critical error occurs.

Survival time The length of time a system continues to function correctly before experiencing a failure.

Cross-Validation A method for evaluating model performance by repeatedly dividing the dataset into parts, training the model on some parts and testing it on the remaining parts to ensure generalisation.

Distance A numerical measure of how different or similar two pieces of data are from each other, interpreted geometrically as the space between points.

Metric Space A set in which distances between any two points can be consistently measured.

Linear A relationship between variables that can be described using a straight line.

Linearly Separable A condition where a straight line (or hyperplane in higher dimensions) lie on different sides of this plane.

Kernel A mathematical function that allows computing the inner products in a higher-dimensional space without explicitly determining the coordinates in that space, often used in kernel machines.

Gradient A vector that points in the direction of the steepest increase of a function and is used to guide model parameter updates during training.

Robustness The ability of a model to maintain performance when facing unusual, noisy, or slightly changed inputs.

Reliability The consistency with which a model or system performs its intended function correctly.

Safety-Critical Refers to applications where incorrect model behaviour could lead to harm or serious consequences.

Security The protection of a model or system from unauthorized access, manipulation, or attacks.

Privacy The principle that individuals' personal information—-such as their identity, behaviour, or sensitive attributes—-should be protected and not revealed by the model, either directly or indirectly, through its predictions or outputs.

Generalisation Gap The difference in performance between the training data and real-world data, which indicates how well the model can generalise.

Fairness The degree to which a model's decisions are free of unjust discrimination across different groups or individuals.

Attack A deliberate strategy used to induce undesired behaviour in a piece of software (e.g., a machine learning model).

Defence A method or mechanism designed to protect a model from attacks or to reduce the harm caused by attacks.

Adversarial Input Describes inputs specifically designed to deceive or confuse a machine learning model.

Benign Input Describes inputs that are natural, expected, and not intended to deceive the model.

Contents

Chapter 1

Introduction

Despite widespread adoption [29, 128, 136, 193, 10] and investor exuberance [146] regarding *artificial intelligence* (AI), there are myriad hurdles between the current state-of-the-art and systems that can be legally declared "safe". AI in general is an ill-defined concept [165], in the words of the human-computer-action pioneer, Larry Tesler, "Intelligence is whatever machines haven't done yet" [84]. Therefore, throughout the text, the term *machine learning* (ML) will be used instead, which is a well defined statistical concept [179]. While the terms are used rather interchangeably in the wider literature, even the most advanced large language models (LLMs) "cannot perform genuine logical reasoning" [142] and calling these models "intelligent" overstates their abilities enormously. While machine learning models hold immense promise—such as lowering healthcare costs [58], improving translation across languages [144], and reducing car-related fatalities [129]—their widespread deployment also raises serious concerns. These include unintentional bias [34], misuse for mass surveillance [5], high energy consumption [2], and the potential to exacerbate global inequality [25]. Therefore, a systematic analysis of the "trustworthiness" of these ML systems is required. This project is a systematic approach to measuring the trustworthiness of these systems.

In particular, the following work seeks to categorise and quantify the effect of inputs that causes errors at run-time (attacks) and methods designed to mitigate their effect (defences), primarily in Paper I and Paper III. In addition, methods for measuring and predicting the cost-efficacy of these attacks and defences are also examined in Papers II, III, & IV. A simple and efficient method for minimising the attack surface of a classification model is presented in Paper V. Finally, a framework and software tooling for safety-critical ML verification and validation is outlined in Paper VI.

1.1 Motivations

The primary motivation for this work is the "social nature of software" [213] and a desire to ensure that deployed models are inherently safe not only for their benefactors, but for the wider public. Given the environmental costs associated with modern machine learning models [2], their known tendency to systematise human bias, and the proposed safety-critical applications like medical imaging, industrial control, and driving, it is critical that these models respect the same laws that govern all other electro-mechanical systems. That is to say—ML models should be as safe as one's toaster, car braking system, or aircraft guidance system and should be around safety [90], quality assurance [89], bias [191], security [4, 163], and the ability to be explained, audited, and reproduced [89, 88, 90, 119, 118]. This works first quantifies the risk associated with trusting these systems, formalises the cost benefit relationship between a given attack and defence, presents a model that categorically avoids many of these risks, and provides a safety-critical framework for not only reproducing the experiments in this text, but for any model built using any number of popular frameworks. The goal, then, is to not only demonstrate why this safety-critical mindset is necessary, but to also provide methods and tooling that allow anyone to replicate the process across any domain to ensure that preventable failures are caught before a model is deployed in the real world.

1.1.1 The Trolley Problem

The *trolley problem* is a classic ethical thought experiment intended to raise questions about decision making and morality. The problems describes a trolley that is heading towards a group of five people who are tied to the tracks. However, the problem presents a hypothetical conductor with a choice to divert the train to another track in which only one person is doomed. While this question is clearly designed to be abhorrent, the conductor, in reality, is not hypothetical. The conductor is already-existing software [78, 62, 100] that is designed to drive autonomous personal vehicles [29], long-distance freight trucks [125], or autonomous drones [35]. If we abstract the problem a bit, then the applications of ML to medical imaging [58], industrial control [106], bureaucratic decision-making systems [193], and law enforcement [10] raise similar questions. Furthermore, it has been shown that there's an inherent tension between the model that performs best *on average* and the one that is most accurate for a given (marginalised) subset of the population. ML Models will, at best, obscure biased data behind opaque or hard-to-interpret decisions [34].

The inherent risks to user privacy [5] and questionable ability of ML models to be safely deployed in the real-world [138, 50, 57, 141, 138] pose ongoing problems that have yet to be solved. In 1966, the "Summer Vision Project" was designed to solve the problem of computer image-based classification at the Massachusetts Institute of Technology [159]; however, as recently as 2023, it was clear that ML systems "consistently fail to meet established safety-critical

standards by a wide margin" [138], suggesting that the horison of artificial intelligence is forever in the future. However, the intent of this text is not to be overly pessimistic. Automated systems capable of learning have tremendous positive potential as well—from real-time translation and office-task automation to data-driven agriculture and reduced healthcare costs. To put it simply, models deployed safely have the potential to improve or save human lives. The trolley problem then becomes a useful analogy for discussing the trade-offs between performance, risk, and cost of ML systems and will frame the rest of the text.

1.1.2 Turing Test

Like the trolley problem, the Turing test is a classic thought experiment that is fundamental to the question of machine trust. Devised by famed computer scientist, Alan Turing, in his 1950 text, "Computing Machinery and Intelligence" [195], in which he describes a simple game involving three people, here referred to as Alice, Bob, and Corey (player A, B, C, respectively). Turing imagined Player A to be a woman and Player B to be a man while Player C is tasked with deducing the gender of the other players using only a series of written questions and answers. Turing then poses the question:

> "What will happen when a machine takes the part of A in this game? Will the interrogator decide wrongly as often when the game is played like this as he does when the game is played between a man and a woman?" These questions replace our original, "Can machines think?"

He goes on to refine this question in the same paper, imagining that in addition to player A, player C is also replaced with a computer program:

> Let us fix our attention on one particular digital computer C. "Is it true that by modifying this computer to have an adequate storage, suitably increasing its speed of action, and providing it with an appropriate programme, C can be made to play satisfactorily the part of A in the imitation game, the part of B being taken by a man?"

While this standard has been of little interest to AI researchers in practice [166] and has many theoretical detractors [81, 166, 65], it functions as philosophical concept more than a qualitative metric [70]. Nevertheless, researchers have noted the market- and research-based incentives to *imitate* humans rather than augment their existing abilities, calling this false goal of imitation a "Turing Trap" [32]. Modern models can often outperform humans on specific tasks in which large amounts of data is available and annotated, but as this thesis demonstrates, this is merely statistical imitation that often fails.

1.1.3 Data Feminism

Feminism in the words of famous scholar and feminist, Bell Hooks, is "for all people, female and male, liberation from sexist role patterns, domination, and oppression" [85] and other authors highlight the need to focus on the intersection of categories like gender, class, sex, and race. *Intersectionality* is "the means for dealing with other marginalisations as well." [49]. "Data feminism", is a book by Catherine D'Ignazio and Lauren F. Klein [53], but the framework it provides for thinking about the power relationships with data in the modern world can also be called *data feminism*. The central conceit of the book is something that is widely forgotten in the world of data science—that power and data are distributed unequally. This is highlighted famously in a pair of studies by Joy Buolamwini *et al.* [34, 167].

Buolamwini's first study (2018) [34] highlighted the fact that commercially available facial detection products had discrepancies in performance across race and gender dynamics with models performing best on white men and worst on women of colour. This study was so impactful that by 2019, these discrepancies had largely been solved, but not erased entirely [167]. However, even in 2023, neither Google's nor Apple's photo app would even attempt to identify gorillas using their respective object detection methods due to a 2015 incident that confused black humans for gorillas, perpetuating a racist stereotype [74]. The power dynamics inherent to modern ML techniques not only have the potential to perpetuate racist and sexist biases, but also rely on extracting data from users often using dubious amounts of consent [5], process it using low-wage workers [39], and consume massive amounts of power [2] to the exclusive benefit of people in the developed world [25].

While this thesis does not focus the disparities in performance between demographic groups, it does examine the effect of changing a sample by a small amount. While many of the techniques and metrics discussed throughout the thesis can be applied to examine the effect of gender or race on model performance, the describes methods can be used in any domain that seeks to verify the performance of a model. As is demonstrated throughout this text, modern ML models fail to meet any legal requirements for safety, regardless of demographic. Therefore, a framework for critically evaluating any ML pipeline through the lens of machine trustworthiness is necessary. This thesis first attempts to quantify the problem before using the measured data to suggest some fundamental solutions.

1.2 Research questions and problem

In this thesis, we attempt to address the following research questions, each of which has a corresponding paper, listed in order.

R1 To what degree can we trust modern ML models?

R2 Can we overcome an adversary by generating or using more data?

R3 How can we formalise the cost benefit relationship between robustness and particular choice of model?

R4 Can we extend the analysis in R3 to examine the relationship between cost and robustness?

R5 Is it possible to build a classifier that circumvents the inherent weaknesses of centralised training paradigms?

R6 How can we build auditable and reproducible tests to meet regulatory standards for safety critical systems?

1.3 Methodology

The generic template for a machine learning project is quite routine. After the data has been collected and labelled, it is often split into "training" and "testing" sets. The training set is used to learn characteristics of the data that can be used to make a decision on unseen (or testing) data using some (simplified) mathematical approximation of the real world (called a *model*). The central assumption is that performance on this test data will generalise to all unseen data. As such, test set accuracy is often used as a way to compare and benchmark models [52, 204]. However, it has been shown many times that small changes in model input can result in drastic changes to the model output [42, 71, 24, 36, 145, 42]. If these perturbations are added intentionally, they are referred to as *attacks*. Techniques to mitigate the effect of these perturbations are called *defences* [209, 71, 214, 117]. By using attacks as a way to simulate worst-case scenarios, a model's *robustness* to these attacks can be measured. By comparing the relative success rate of an attack in response to various model defences, we can determine the relative efficacy of a defence or attack. To evaluate these attacks and defences in practice, a variety of benchmark datasets were employed throughout this work in this thesis, each chosen to highlight specific challenges or characteristics relevant to adversarial robustness.

1.3.1 Data

In this thesis, two types of datasets were examined—broadly classed into images and tabular data. The image datasets include Modified National Institute

of Standards and Technology (MNIST) database [55] that includes 60 thousand images of low-resolution, hand-written digits as well as the Canadian Institute for Advanced Research 10 category and 100 category databases [111] (CIFAR10 and CIFAR100 respectively) that contain tens of thousands of higher resolution colour images that depict objects like airplanes and cats. These image databases have become a de-facto standard for measuring model performance and they have existed for decades [55, 111]. MNIST in particular is considered relatively easy because it uses low-resolution black and white images and only a small number of distinct classes (one for each digit). In addition, CIFAR10 is known to be easier than CIFAR100 [179, 57] despite the two databases having images of the same resolution and colour depth as the difficulty of a classification increases with the number of distinct categories. Several standard anomaly detection datasets were also evaluated. The oldest such dataset is Knowledge Discovery and Data-mining (KDD) database that was cleaned and dubbed the network security laboratory (thus KDD-NSL) [189]. It captures system-level data that tries to capture the behaviour of both normal (benign) users and malicious software (viruses, malware, etc.). Similarly, the distributed denial of service dataset (DDoS) provided by the University of Toronto was also of interest [151] as it captured network-level data that tries to capture the behaviour of both normal users and a variety of attack types. Both of these datasets distribute the data with several class labels to distinguish between different types of attacks; however, for the purposes of this thesis all attacks were collapsed into a single *adversarial* category. However, KDD-NSL has a only a few thousand examples and DDoS has 10s of thousands as the origins of these datasets are separated by decades. This is the natural division of the next two datasets—the short message system spam (SMS spam) database [8] and a database of tweets, metadata, and "bot-scores" collected from the platform formally known as Twitter [102] (dubbed Truthseeker). These two datasets both contain benign and adversarial user data and are widely used for the problem of *anomaly detection* that attempts to broadly classify data into "normal" and "abnormal" categories. In Paper II, in an effort to use the best-case dataset, myriad datasets were generated using normally distributed, arbitrary data and the exact details are described in that paper.

1.3.2 Model

ML pipelines are often long-running and complex software tool-chains, often with many parameters that must be adjusted according to the dataset and model type. The set of all knobs, levers, and dials that a model-builder can tweak is said to be the model *hyper-parameters*, where the hyper- prefix is used to distinguish these *tuned* values from values that are *learned*—often called model *parameters* or *weights*. The training set is used to teach the model patterns in the data, which involves finding optimal values for these parameters given a specified set of model hyper-parameters. This score is always calculated using the test set that was withheld during model training in order to test the

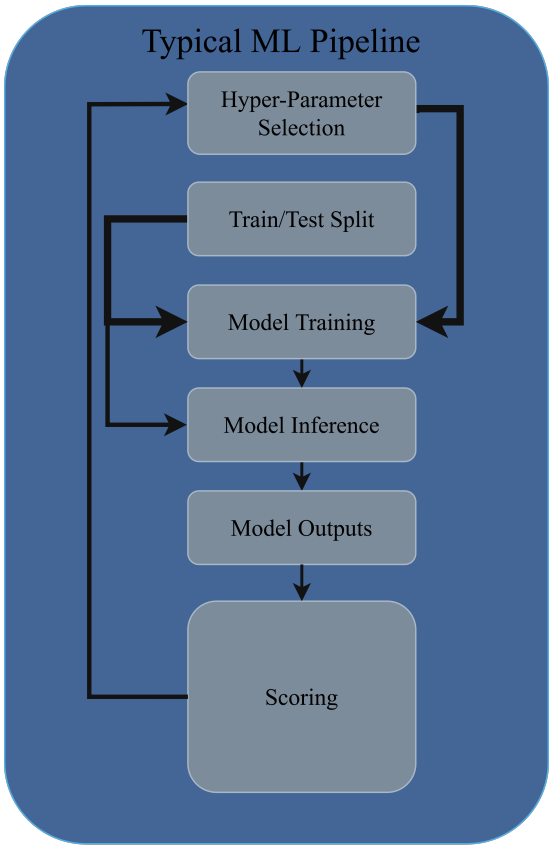


Figure 1.1: Modern models typically have many hyper-parameters that must be configured by a model builder and internal parameters that are configured during model training through a chosen optimisation process and criterion. A dataset is split is split into "train", "test", and sets, which are used to allow the model to learn patterns and evaluate model performance, respectively. Models are repeatedly scored according to the chosen criterion, with the goal of finding the best model for a particular dataset. Another partition of the data, called the "validation" set (not pictured), is used to verify that this "best-fit" model will generalise to unseen data.

ability of a model to generalise to unseen data. If a model performs much better on the training data than the test data, it is said to be *over-fit*. While this modelling process aims to optimise performance on unseen data, it typically assumes that inputs are crafted by users without malice—in practice, however, these models are vulnerable to targeted manipulations known as adversarial attacks.

1.3.3 Attacks

In the context of machine learning, an *adversarial attack* refers to deliberate and malicious attempts to manipulate or exploit machine learning models. Adversarial attacks are designed to deceive or mislead the model's behaviour by introducing carefully crafted input data that can cause the model to make incorrect predictions or produce undesired outputs. The goal of an adversarial attack is often to exploit vulnerabilities in the model's decision-making process or to probe its weaknesses. These attacks can occur during various stages of the machine learning pipeline, including during training, inference, or deployment. Figure 1.2 show how each class of attacks targets a different stage of the pipeline depicted in Figure 1.1. Evasion attacks aim to manipulate input data during the inference phase to deceive the model into misclassifying or ignoring certain inputs. Attackers carefully craft perturbations or modify the input features to mislead the model while still appearing similar to the original input [24, 36, 31, 110, 42]. In poisoning attacks, the attacker intentionally injects malicious or manipulated training samples into the training dataset where the goal is to influence the model's behaviour during training so that it learns to make incorrect predictions or exhibit unwanted behaviours when presented with specific inputs [22, 176]. Inference attacks exploit the model's output or responses to obtain sensitive information about the training data or other confidential details. By observing the model's predictions or confidence scores for carefully crafted inputs, attackers can extract information that was meant to be kept private [38, 156]. Model inversion attacks (also called extraction attacks) aim to infer sensitive information about the training data or proprietary model by exploiting the model's outputs. Attackers utilise the model's responses to iteratively reconstruct or approximate training examples that are similar to the ones used during training [38, 44, 123].

1.3.4 Defences

Defences are techniques that are applied to models to mitigate the effects of attacks. They come in several varieties, classed according to the model pipeline stage the effect. Figure 1.3 depicts how each stage interacts with the typical machine learning pipeline, described in Figure 1.1. This might include adding some Gaussian noise to the data (*pre-processing*) or model output (*post-processing*) with the expectation that the model will be less sensitive to adversaries. If the model-builder changes the input data (e.g., by adding or removing noise), then

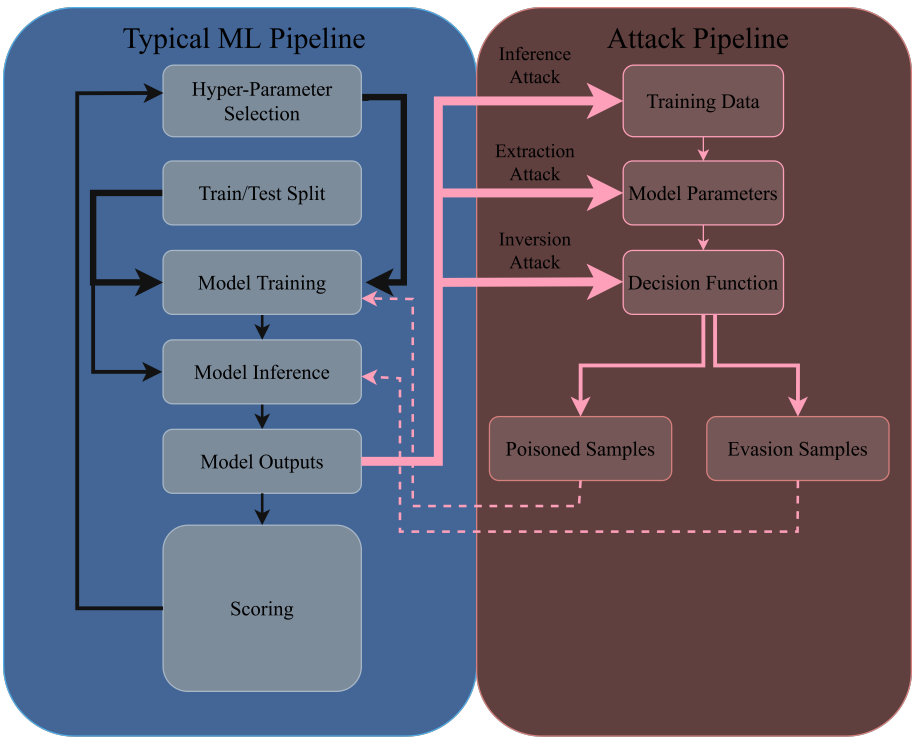


Figure 1.2: A typical machine learning threat model can be attacked in many ways. An attacker might use an inference attack to discover the training data or properties of it. An attacker can use an extraction attack to reverse engineer model parameters and use an inversion attack to reverse engineer the decision boundary with an entirely different model. Any of these attacks can then be used to generate malicious samples that target the model training process (poisoned) or malicious samples that attempt to fool the model during the inference stage (evasion samples), though techniques exist that can exploit particular model APIs (white-box attacks) or only the model output (black-box attacks). The dotted line holds no special meaning and is merely intended to distinguish the arrows that overlap.

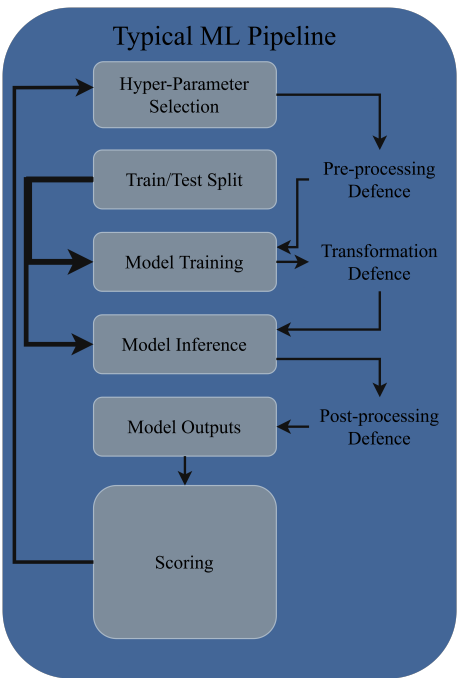


Figure 1.3: A typical machine learning model can be defended against an attacker in several ways. While these techniques are not always applied during the hyper-parameter tuning stage, their effect on the end user must nonetheless be considered. A retraining defence might include any of these strategies in addition to using attacks to generate samples. One can thing of Figure 1.2 as the pipeline needed to conduct a retraining defence, assuming one labels and scores the attacked samples appropriately.

this is called a preprocessing defence. If the modeller changes a trained model with the expectation of more robust inference, then that is called a transformation defence. A post-processing defence is when raw output of model inference is changed before being presented to a user. Alternatively, known adversarial examples can simply be labelled appropriately and incorporated into a new generation of training data. Another technique, primarily discussed in the context of large-scale machine learning methods is model transformation defences that seek to reduce model's sensitivity to input noise [158, 66, 202]. The primary downside of these transformation defences is cost as it scales with both the size of the model and the number of training samples. Other techniques try to detect malicious samples at run-time by examining the inner-workings of the model [41] or by rejecting samples by examining subsets of the training data [130, 150]. The final category is adversarial retraining which uses attacks to create synthetic data before retraining the model on a dataset that includes the generated attacks. Paper II is focused on this technique and explains why it is not a feasible solution for models that are themselves expensive to train. Several of these statistical defences are explored in Papers I-IV, while Paper V discusses a categorically different approach to reducing adversarial risk.

1.3.5 Metrics

Adversarial vs. Benign Performance

Models are often evaluated under both benign and adversarial scenarios. The *benign* scenario is the typical ML process wherein test set accuracy is intended to reflect the probability of real-world success. The *adversarial* scenario occurs when an attacker changes the test or train samples to influence the performance of a model. A model is said to be *robust* if the gap between the adversarial accuracy and the benign accuracy is small [37]. The focus in this work was evasion attacks as they simulate what happens when noise is added to a sample. This noise is intentionally designed to reduce the accuracy of a model by crafting small changes in the model input (generally assumed to be imperceptible to humans) that nevertheless induce large changes in the model output. That is, the benign and adversarial scenarios define the best- and worst-case scenarios for real-world usage. In addition to the benign and adversarial accuracy discussed here, several other metrics were also collected.

Measures of Success and Failure

The primary metric of concern throughout this thesis is the probability that a model correctly predicts an output. In the case of classification systems, the standard metric is accuracy. It is defined as ratio of correct predictions to the total number of predictions, such that

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} = 1 - \frac{\text{Incorrect Predictions}}{\text{Total Predictions}}.$$

Accuracy is used throughout the text as a measure of reliability as it a standard used in the literature to compare the efficacy of published models on benchmark datasets [50]. However, even the weakest of safety-critical regulations [88] would require millions of samples to verify a model using accuracy as a measure of failure probability. For life-and-death applications like medical imaging and self-driving cars, these verification procedures would require many many billions of samples [88]. It is simply infeasible to use such large datasets to verify every software change in complex ML pipelines, as required by regulatory risk management standards [88]. Instead, Paper III formalises the relationship between the probability of failure and time–referred to throughout the text as *failure rate*:

$$\text{Failure Rate} = \frac{\text{Incorrect Predictions}}{\text{Time Interval}} = 1 - \frac{\text{Correct Predictions}}{\text{Time Interval}}.$$

Like accuracy, this can be thought of in both the adversarial and benign circumstances, but first some time intervals must be defined. The *training time* is the time it takes to fit a model with a given set of parameters to a particular dataset and is calculated using the training set. The *prediction time* is the time it takes to receive output from a model given a set of samples and is calculated using the test set. The *attack time* is the time is takes to generate adversarial examples and is calculated using the test set throughout this text. The benign failure rate can be thought of as:

$$\text{Benign Failure Rate} = \frac{\text{Incorrect Predictions on Benign Data}}{\text{Prediction Time}}$$

and the adversarial failure rate can be expressed as:

$$\text{Adversarial Failure Rate} = \frac{\text{Incorrect Predictions on Adversarial Data}}{\text{Attack Time}}.$$

The specific notation used for this concept varies throughout the thesis, but the concept is the same—the benign and adversarial failure rates are used to define an upper and lower bound of model performance that considers both efficacy and cost. The point of collecting these time-based metrics is to consider the cost for both the model-builder and the attacker. In Papers III & IV, an additional metric is considered, called time to failure. It can be defined as:

$$\text{Time-to-Failure} = \frac{\text{Total Time}}{\text{Number of Failures}}.$$

We can also discuss this under specifically adversarial constraints, where

$$\text{Time-to-Failure} = \frac{\text{Total Attack Time}}{\text{Number of Failures}}$$

quantifies both the number of failures and the time required of an attacker. While these metrics provide a practical lens on model performance under real-world constraints, they do not account for the underlying computational complexity that governs how these systems work in real-world deployments. To

understand the feasibility of training, attacking, or defending machine learning models at scale, it is necessary to introduce a formal framework for measuring computational cost.

When computer scientists discuss the notion of cost, it is common to use "big-O" notation where, for two functions $f(x)$ and $g(x)$:

$$f(x) = \mathcal{O}(g(x)) \text{ as } x \to \infty$$

if for some positive constant multiple M and a real number x_0

$$|f(x)| \leq M|g(x)| \text{ for all } x \geq x_0.$$

The notation describes how $f(x)$ and $g(x)$ behave similarly as x grows large. Iterating over a list of length n is said to have complexity $\mathcal{O}(n)$. Likewise, pairwise operations over two lists of lengths m, n yield complexity $\mathcal{O}(mn)$. In modern cloud deployments, computational time is billed by usage and better hardware costs more per unit time, making cost analysis critical to feasibility.

Cost is an essential consideration of this thesis. Paper I examines the efficacy of various model defences in relation to their cost by quantifying the benign and adversarial failure rate of many attacks and defences. Paper II illustrates an attacker's time-cost advantage under certain constraints by comparing the benign and adversarial failure rate when adversarial retraining is employed as a defence. Paper III demonstrates the triviality of attacking a particular and popular ML model by introducing a novel metric that quantifies the relative cost of model-building compared to the cost of a successful attack. Paper IV demonstrates efficacy of using certain low-cost hardware to not only reduce the computational footprint, but to also induce more robust models. Paper V introduces a distance measure that is motivated by the need to develop client-side models that do not rely on widespread data surveillance.

While cost considerations determine the practicality of deploying and maintaining machine learning systems, legal and regulatory requirements must also be considered—especially against the backdrop of adversarial attacks. The next chapter addresses the regulatory standards that govern safety, privacy, and security, outlining the constraints under which trustworthy and responsible AI *must* operate.

Chapter 2

Regulatory Concerns for Trustworthy ML

Safety critical systems are not new. We have been successfully conducting trans-oceanic flights for more than a century and successfully visited the moon. Even though billion of flights happen every year, passengers expect to arrive at their destination safely. Potentially dangerous devices like lithium batteries, nuclear power plants, and toasters power billions of devices, homes, and people without incident on a daily basis. This is not due to random chance or some invisible hand of profit motive, but decades of regulations and engineering. For example, even though car crashes cause more than one million deaths around the world every every year [157], less than 1 in 10^{-8} fatal accidents occur per kilometre driven (assuming estimates from the United States [154] hold globally). The National Highway and Transportation Administration in the United States estimates that only 7% of all crashes in the United States are due to deficiency in the car and 70% of that is attributed to degraded tires [185]. Road design is only considered a factor in 1.6% of cases [185]. That is to say, the standards that govern the engineers behind car braking systems and road design are clearly working. In the United States, many of these standards are defined by the National Institute of Standards and Technology (NIST), which is a US federal government agency [1]. The International Standards Organisation (ISO) and the International Electrotechnical Commission (IEC) are non-governmental organisations, headquarted in Switzerland, that develops standards globally with the latter obviously focused on electronics and software. These three organisations govern safety-critical products across many domains— medical devices, industrial equipment, and consumer products designed for all manner of tasks.

These regulations are discussed in the context of safety, security, privacy, and reliability before examining the specific requirements to verify and validate these properties. In the broader context of this thesis, which aims to advance

the trustworthiness of machine learning systems, it is essential to recognise that ML models—particularly those deployed in safety-critical domains—must be held to *at least* the same rigorous standards that govern traditional technologies such as kitchen appliances, mobile phones, and auto-mobiles. Without such standards, the deployment of ML systems in safety-critical areas will cause catastrophes. Below, the already-existing regulations around safety, security, privacy, and reliability are the discussed in the context of trustworthy machine learning.

2.1 Safety

Safety is a well-defined regulatory concept, particularly for electronic hardware and software, where it is governed by the IEC 61508 standard [88]. It is the de facto global framework for safety-critical design of electromechanical systems [147], though domain-specific equivalents (*e.g.*, healthcare [119] or anti-virus software [163]) exist as well.

To understand how regulatory limits are set, IEC 61508 first defines failure rate categories (Table 2.1). These numbers may appear extreme, but they are appropriate when designing products with millions or billions of users. For instance, the bare minimum standard could be one preventable death per year per 10 million users. In the context of billions of drivers or commercial airline passengers, 1 failure in 10 million still means thousands of foreseeable and preventable and potentially fatal accidents.

The standard defines "frequent" failures as those occurring more than 10^{-3} times per year—meaning multiple times within an individual's lifetime. Crucially, the severity of the failure also matters. Terms like "catastrophic," "marginal," and "negligible" are precisely defined (Table 2.2). IEC 61508 then introduces Safety Integrity Levels (SILs)—four classes based on risk, combining failure rates and consequence severity (see Tables 2.3 and 2.4). Together, they establish a risk matrix that balances the likelihood of a failure with its impact.

Table 2.1: Failure Rate Categories according to IEC61508.

Category	Definition	Range (failures per year)
Frequent	Many times in lifetime	$> 10^{-3}$
Probable	Several times in lifetime	10^{-3} to 10^{-4}
Occasional	Once in lifetime	10^{-4} to 10^{-5}
Remote	Unlikely in lifetime	10^{-5} to 10^{-6}
Improbable	Very unlikely to occur	10^{-6} to 10^{-7}
Incredible	Cannot believe that it could occur	$< 10^{-7}$

The IEC standards [88] define the Safety Inegrity Level (SIL) in failures per year, which is reproduced here in Table 2.3. Since safety-critical systems (e.g., autonomous vehicles, medical imaging software, malware detection systems,

Table 2.2: Consequence Categories according to IEC61508.

Category	Definition
Catastrophic	Multiple loss of life
Critical	Loss of a single life
Marginal	Major injuries to one or more persons
Negligible	Minor injuries at worst

Table 2.3: Risk Classes according to IEC61508.

Class	Range (failures per year)	Definition	Failure Rate Category
I	$[10^{-6}, 10^{-5})$	Unacceptable	Occasional
II	$[10^{-7}, 10^{-6})$	Undesirable	Remote
III	$[10^{-8}, 10^{-7})$	Tolerable	Improbable
IV	$[10^{-9}, 10^{-8})$	Acceptable	Incredible

etc.) have the potential to cause major injuries (or worse), validating the failure rate using a test set would require at least 10s of thousands of samples and highly confident measures would require substantially more. Furthermore, if we assume that *accidental* errors are possible in real-world systems due to things like dust, lens flare, component failure, packet loss, *etc.*, it naturally follows that the failure rate is an over-estimate of the model's performance at the edge or other "worst-case scenarios". That is, the test-case failure rate is an optimistic overestimate of the real-world failure rate [12].

2.2 Security

Like safety, security is subject to well-defined standards. These address both organisational practices and technical mechanisms. For example, the ISO/IEC 27000-series focuses on information security management systems, helping organisations ensure secure development and operations by setting the best practices for *threat modelling*, which is the process of outlining risks and specifying mitigation techniques (*e.g.*, attacks and defences respectively). NIST Special Publication 800-53 [147] provides a catalogue of security and privacy controls, similar to the ISO27000 series of regulations. IEC 18003 defines standards for cryptographic techniques, encryption algorithms, key management, and hashing. Notably, ML models are generally not considered cryptographically secure [5], which raises profound concerns about data and model privacy [147] as these models are known to leak not only information about themselves [147] but also information about the data used to train models [147].

Both the ISO and NIST frameworks aim to secure safety-critical systems from malicious actors, ranging from lone attackers to nation-state-level threats.

Table 2.4: Consequence vs. Risk Matrix according to IEC61508.

Likelihood	Catastrophic	Critical	Marginal	Negligible
Frequent	I	I	I	II
Probable	I	I	II	III
Occasional	I	II	III	III
Remote	II	III	III	IV
Improbable	III	III	IV	IV
Incredible	IV	IV	IV	IV

Just as we analyse system failure rates for safety, we must assess threat models and attack surfaces for security. Proper cryptographic implementation, access control, secure coding practices, and real-time anomaly detection all play vital roles. Notably, both ISO and NIST standards discuss malware as software that exfiltrates or destroys user data without the consent of the user. Without a proper understanding of the risks of deploying a model, it is simply not possible to give informed consent. As such, ML systems that rely on consent from ill-informed users about their safety and efficacy can rightfully be considered privacy-invading malware.

2.3 Privacy

Privacy standards vary significantly throughout the world; however, some commonalities remain.

In the United States, privacy regulations are dominated by the Children's Online Privacy Protection Act (COPPA) and the Health Insurance Portability and Accountability Act (HIPAA) govern the data privacy rights of children and healthcare recipients respectively. COPPA clearly defines mechanisms for consent in which privacy policies must enumerate collected data, the usage of the data, and to whom the data is shared. Under COPPA, parents must be able to review, delete, and revoke consent for continued use of their child's data. Under both COPPA and HIPPA, data security (governed by ISO-27000 or NIST-800) is a requirement and breaches are the liability of the entity that holds the data. This security requirement is upheld under HIPAA for healthcare providers. In addition, HIPAA mandates the de-identification of patient data, the use of encryption, audit trails and access logging, and a formal method for risk assessment and breach notification [119]. As with COPPA, the relevant users retain the right to access, amend, and receive a copy of their data [118, 119].

In the European Union, data privacy is governed by the General Data Protection Regulation (GDPR) and ML models in particular must comply with the European Union's AI Act if they intend to act within that market. These laws are particularly relevant in safety-critical domains where personal data—such

as identity, health records, location, or behaviour is recorded *or indirectly inferred*. The GDPR grants additional user rights including the right of users to have granular and consensual control of their data and extends these rights to all residents of the European Union, regardless of age or illness. For ML-specific tasks the AI Act applies, which additionally regulates commercial models in various domains, by banning domains that present an unacceptable risk (*e.g.*, social scoring, predictive policing, or real-time biometric surveillance in public) and setting legal standards for other categories of activity. The AI Act designates high-risk activities as thing like medical devices, hiring systems, and critical infrastructure and how they must comply with existing regulations (*e.g.*, the IEC or ISO standards outlined above). The act designates limited risk applications include things like biometric models, AI generated, and sentiment analysis and must disclose the use of AI, must minimise the effects on vulnerable groups, and must make the content accessible to those with disabilities.

While privacy regulations such as GDPR and sector-specific laws like HIPAA or COPPA define how personal data must be collected, stored, and processed, they are largely silent on the reliability of the systems that process that data—particularly when those systems involve ML. However, as AI models increasingly influence decisions in domains such as healthcare [58], security [10], transportation [29, 128], and criminal justice [193]. A system that preserves privacy but produces unpredictable or erroneous outputs can still cause significant harm.

2.4 Reliability

As ML systems continue to be integrated into safety-critical domains, their reliability has become a central engineering concern even if the aforementioned legal regulations lag behind. Unlike traditional software systems, ML models are inherently statistical in nature and their behaviour is shaped not just by the complied code but the data set, sampling method, training procedure, architectural choice, and underlying hardware. This seeming lack of reliability creates an essential challenge—even if a model performs well *on average*, we do not know how it will perform outside of the laboratory-based data-sets (*e.g.*, degraded sensors, lens flare, dust, or statistically marginal subsets of people). Since the number of possible failure cases is often innumerable, it is essential to define the threat model and failure conditions for an ML model. In the literature, these failure conditions are often considered under various worst-case scenarios that are designed to be adversarial. A model is said to be *robust* if it resists these adversarial failures.

For example, we could see how different AI-pipelines influence the accuracy of a given model architecture and dataset against various adversarial conditions or train a model to optimise for loss in the presence of adversaries [50, 122]. Even within adversarial machien learning publications, research has con-

sistently been shown to be optimistic and unreliable at best [52].

Paper I presents a comprehensive survey of various model attack and defence techniques, demonstrating that, despite their widespread use in the literature [138], commonly reported robustness metrics do not reliably predict performance against a generalised adversary. Paper II investigates the impact of minor hyper-parameter variations and shows that naive adversarial retraining tends to degrade model performance under different attack scenarios, while also introducing significant (polynomial-time) computational overhead. In the broader context of complex electro-mechanical systems, modelling and mitigating failures or unforeseen edge cases is typically addressed through systematic verification and validation processes [90]. Then the cost-robustness trade-offs are examined in the context model complexity (Paper III) and hardware choice (Paper IV), before attempting to design a model that is safer-by-design (Paper V). Finally, a safety-critical framework for ML verification and validation is presented in Paper VI.

Chapter 3

Machine Learning

Machine learning is a branch of mathematics and computer science that attempts to build models that can learn from data and/or generalise to unseen data to perform a task without explicit instructions. For a full treatment of statistical learning theory, see "Understanding Machine Learning" [179]; here, a subset of models is introduced to illustrate key considerations and limitations. Figure 3.1 provides a visualisation of the models discussed below and their efficacy on different types of data.

3.1 Linear Regression

Linear regression is one of the simplest and most widely used models for supervised learning tasks that outputs a continuous value (called *regression*). While the other models in Figure 3.1 are classifiers, the "classes" shown in that figure designate samples with an output $> .5$ as red and others as blue to allow for the comparison of this basic model with the others specified below. It models the relationship between a scalar dependent variable and one or more independent variables by fitting a linear model to the observed data. However, there are certain disadvantages to this approach. Linear regression assumes that the relationship between the features and the target variable is, in fact, separable by a line, which does not hold in many real-world scenarios. Despite its limitations, linear regression serves as a foundational model in machine learning, providing a solid basis for understanding more complex techniques.

3.2 Logistic Regression

Logistic regression is a widely used classification method for binary outcomes, modelling the probability that a given input belongs to one of two classes. Rather than producing an output that corresponds to a value in the range of $(-\infty, +\infty)$ (as in the case of linear regression), it produces an output in the

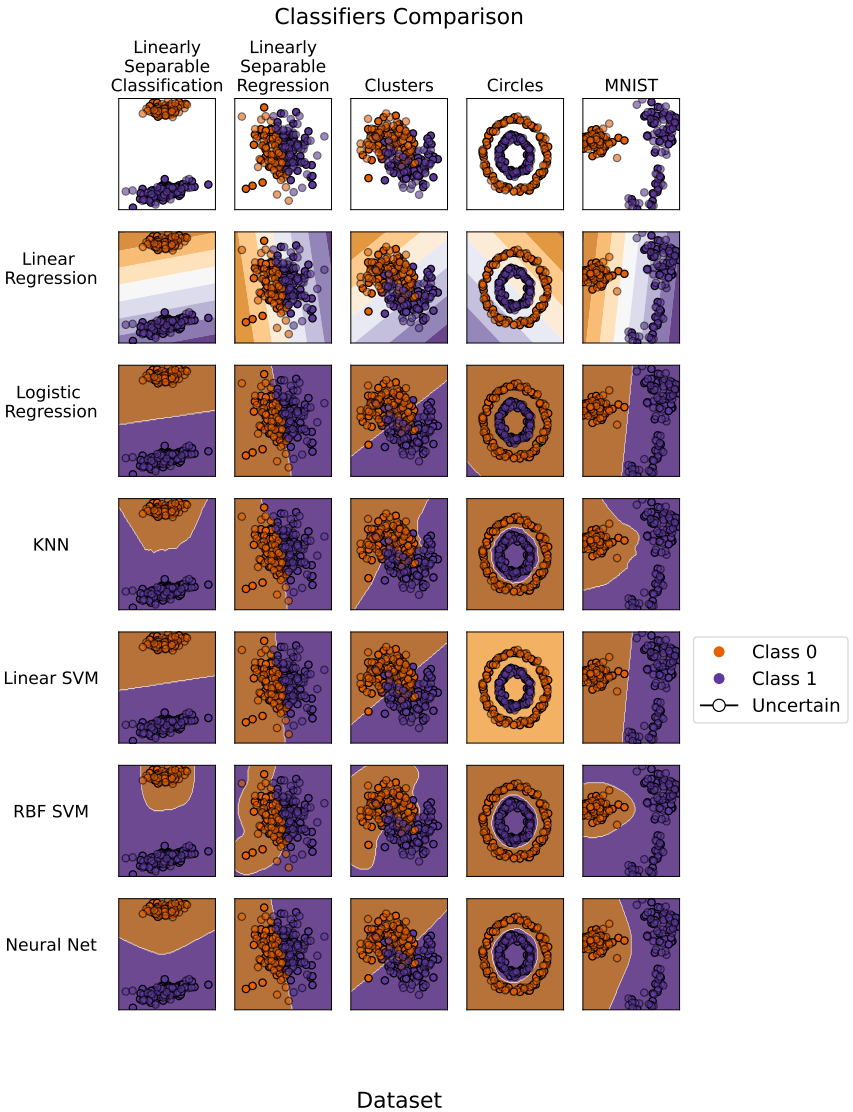


Figure 3.1: Each row depicts a different dataset where the values are classified into one of two classes, depicted by purple or orange dots and each column depicts a single dataset. The model's *decision boundary* is depicted using the purple and orange gradients with darker colours corresponding to a more confident decision. The first row depicts the raw input data and the true labels. Each other row depicts the prediction using colour. The first 4 columns used generated two-dimensional data and the last row depicts a two-dimensional projection of the MNIST image dataset discussed in Section 1.3.1 (since it would be impossible to plot the data in the 784 dimensions MNIST provides in the form of black and white pixels).

range $(0, 1)$, which can be interpreted as a probability. Logistic regression uses a sigmoid (logistic) function to map the model output to the range , making it ideal for likelihood estimation.

In short, a logistic regression model produces an output that answers the question: "what is the probability of a belonging to a class when we assume a linear decision boundary?"

3.3 KNN

Another model used for classification is k-nearest neighbours (k-NN), which is a type of instance-based learning where a data point is classified based on the consensus of its k closest neighbours according to some relevant notion of distance (*e.g.*, the "edit distance" for strings or some number of pixels for images). For classification tasks, that might me mean finding the class that belongs to the most neighbours and for regression task that might mean a simple average. This model is widely used for classification and regression tasks, and in Paper V, it was specifically employed to cluster spam and non-spam internet traffic and content, for instance.

3.4 SVM

Support Vector Machines (SVM) are a powerful class of supervised learning models used for classification and regression tasks. The core idea of SVM is to find a surface (decision boundary) that separates data points of different classes in a high-dimensional space in such a way that the distance between the classes is maximised. While the linear SVM relies assumptions about the linearity of the data, versions using kernel functions allow it to capture complex or high-dimensional relationships in the data. Through kernel functions, SVMs can create complex, non-linear decision boundaries. The decision boundary generated by SVMs, particularly with non-linear kernels, can be difficult to interpret [179]. These kernel SVM models are used in Papers II and V.

3.5 Neural Networks

Neural networks are another class of ML models that can be both parametric [198] and non-parametric [164]. A neural network can be viewed as a computational graph composed of layers of interconnected nodes. It learns representations of the input data in multiple stages: each hidden layer extracts features from the output of the previous layer, progressively building more abstract or complex representations. The final layer typically maps this learned feature representation to an output using a standard machine learning model, such as a linear classifier or regression function. They often return a vector of

probabilities or likelihoods (called a *soft label*), or a series of binary classifications (called a *hard label*). Figure 3.2 depicts a simple neural network using circles to represent nodes, black lines to represent edges, and colour is used to identify input, hidden, and output layers.

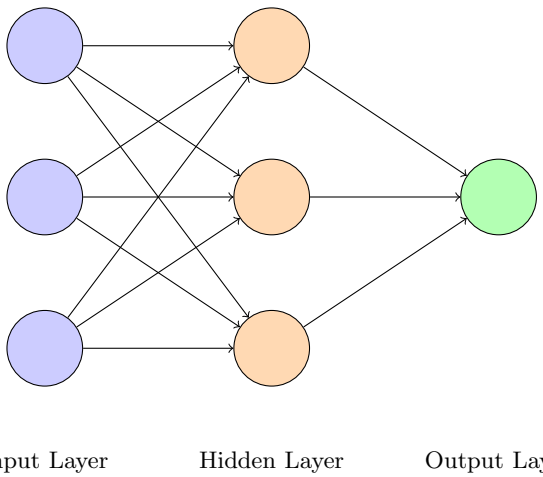


Figure 3.2: A simple neural network with three layers. The input (blue), hidden (orange), and output (green) layers are colour-coded for clarity. The arrows indicate the flow of data.

Typically, state-of-the-art models are significantly more complex than the one pictured in Figure 3.2 and often rely on large numbers of hidden layers, complex structures (also called *architectures*) that, for example, provide mechanisms to reinforce a certain behvaiour [101] or enable time-dependent model configuration [83]. Model architectures are sometimes chosen in an effort to model a particular biological [17] or physical process [14], though model architectures are often determined by brute-force trial and error [21].

3.6 Loss

Regardless of the model, training it requires the definition of a loss function—a quantitative measure of how far the model's output deviates from the expected result. The choice of loss function is often dictated by the task at hand: for example, the cross-entropy loss is widely used for classification problems [179] because it naturally penalises confident but incorrect predictions; whereas mean squared error (MSE) [115] is commonly applied in regression tasks where the goal is to minimise the average squared difference between predicted and actual values. More sophisticated loss functions can encode domain-specific requirements, such as robustness to outliers [217], uncertainty visualisation [218], or

fairness constraints [132], and can be custom-designed to reflect complex trade-offs or safety margins.

Furthermore, in domains such as healthcare, aviation, or autonomous systems, loss functions and optimisation criteria may be carefully designed to reflect asymmetric costs of error [139]. For instance, this might mean penalising false negatives more heavily than false positives when a missed detection could result in harm than a false positive. In other cases, this might mean providing mechanisms to notifying a user when a given classification is uncertain [218]. Nevertheless, even an excellent score on well chosen metric does not guarantee legally compliant performance in the presence of an adversary [139, 137, 140, 141], which are characterised further in Chapter 4. In addition to a well-chosen loss function, the optimisation algorithm must also be robust.

3.7 Optimisation

The optimisation process involves adjusting the model's internal parameters—typically numbered in the millions or even billions—to minimise the chosen loss function across a given dataset. This is most often done using stochastic gradient descent (SGD) or one of its many variants (e.g., Adam [105], RMSProp [222], or Adadelta [215]) which iteratively updates model parameters in the direction that minimises the chosen loss function by comparing model predictions to the ground truth.

The optimisation landscape of many ML models is highly non-convex—it often contains numerous local minima, saddle points, and flat regions. In practice, modern optimisation algorithms, combined with techniques like learning rate scheduling [109], regularisation [208, 93, 173], momentum [188], and batch normalisation [177], enable effective navigation of this complex terrain. However, convergence to a low loss value does not necessarily imply generalisation to new data. As a result, the optimisation process is closely tied to the choice in validation metric [37] and techniques for measuring generalisation error like cross-validation scoring [179] are used to ensure that the model not only performs well on training data but also works well *in general*.

3.8 The Ethics of ML

The problems with the typical approach to ML are numerous. Recently, marginal gains in accuracy have relied on increasingly larger models and datasets [56]. These models rely on massive datasets [56, 200, 28], from an ever-shrinking number of institutions [107], leading to gender-biased models [126], race-biased models [34], and critical design errors [16]. This trend goes back decades [48, 168, 34]. Despite this, it has been suggested that large scale ML methods be used to detect (*potentially*) illegal material [5].

Since client-side monitoring with neural networks was recently the subject of a proposal by the EU parliament to stop the spread of offensive and harm-

ful online contact [59], it can be used as a simple case study in how *obviously* "good" models have inherent risks. First, these models are not nearly reliable enough to be deployed in situations that can negatively effect a human, which is detailed in the previous chapter. Second, these models can reveal the private data used to train the model [92] or used to generate novel content that is similar to restricted images [220]. Third, these models are inherently biased—relying on statistical assumptions that harm already marginalised groups [34]. Fourth, the existence of the model comes with unforeseen environmental and societal costs—potentially chilling dissent or speech [5], using immense computational resources [2], and outsourcing of the data labelling to low wage workers who have to view the offensive content [25]. Finally, modern ML systems in general have questionable reliability on edge cases, but nevertheless induce real experts to misjudge the quality of the model output [134, 78, 62, 100, 162, 172]. In short, there are many ethical questions to consider when deploying an ML model which have been clustered in this Chapter into reliability, reproducibility, fairness, bias, deferred expertise, and the environment.

3.8.1 Reliability

Fundamentally, ML systems are just maximum likelihood models that are prone to over-fitting. ML models have shown themselves capable of performing exceptionally well on a wide variety of tasks. However, much research has also noted the fragility of these models to small changes in the input space [71, 36, 127, 145, 42, 145, 110]. One particular concerning example is the inability of state-of-the-art image classification systems to handle data that comes under real-world conditions that vary from training set [33, 12]. One proposed source of this error is "hidden stratification"—the unknown subsets of the data that are unaccounted for in the model [155]. Another often-discussed source of unreliability is the "long-tail problem" that describes the problem of the wide distribution of real-world cases outside of the distribution learned from the training set [219]. A 2021 survey [108] of wildlife identification models examined methods for mitigating the effect of these out-of-distribution new samples and their best attempt still confidently returned an incorrect answer nearly 30% of the time. Another author notes that many medical models do not convey uncertainty to users, thus leading users to overestimate the models' reliability [7].

3.8.2 Reproducibility

Unlike the experiments detailed in the included papers, many published ML papers are not reproducible [76, 87, 46, 77, 143]. Firstly, there is a problem of scale as ML research has increasingly shifted to industry due to the massive costs associated with training state-of-the-art models and employing the people capable of doing so [6, 99]. This is in no small part due to the scale of both modern models and their associated datasets [56, 2] which have grown in

size [56]. For example, commercially deployed image recognition systems and large-language models can cost millions of dollars of compute time to train [148], making the reproduction of state-of-the-art research feasible for only a small number of organisations.

Even when model architectures are fully released, critical components can be omitted or under-specified—including details about training schedules, hyper-parameter selection, hardware or network configuration, and random seed initialisation [87, 46, 77, 143, 76]. Furthermore, many proprietary models are never released at all and the training data remains entirely private or protected by copyright, creating a performance gap between academic benchmarks and real-world implementations [6, 12]. Without verifiable and consistent performance it is impossible to determine efficacy or establish accountability [76]. Reproducibility is a prerequisite for trustworthy, reliable ML systems and dataset and pipeline transparency have recently become the subject of regulatory concern [190, 191].

3.8.3 Bias

Some authors [26] recommend a set of ethics that centres machine learning research on groups mostly likely to be adversely effected. However, since these groups tend to be marginalised or statistical outliers compared to dominant groups, since these groups may represent only a small proportion of the training data, standard machine learning optimisations that minimise average loss naturally focus on the dominant majority, exacerbating disparities in safety and service [34].

ML systems can inherit and amplify biases present in their training data or in the design decisions made by human creators [33, 167]. These biases may manifest as systematic errors or unfair outcomes, such as unequal treatment across demographic groups, because the models optimise for patterns present in the data without an understanding of fairness or ethics [153, 153, 47]. For example, crash safety models historically based on the 50th percentile male body standard have resulted in vehicle designs that provide significantly less protection for female-bodied individuals, contributing to higher rates of injury and fatality for female-bodied people in car accidents [60]. In ML, researchers found neural networks that unintentionally detect self-reported racial information through methods they were unable to explain "either by spatial location, in the frequency domain, or that were caused by common anatomic and phenotype confounders associated with racial identity" [69]. Nevertheless, the problems of bias are not intractable. After raising awareness about the racial and gender discrepancies in commercially available facial recognition systems [34], subsequent research found that the discrepancies substantially reduced by the next year [167], perhaps with nothing more than changing the sampling method [175] (though we can't be sure because the models in question were closed source and then largely removed in the wake of "Black Lives Matter" protesters in the United States [79]. However, computer science as a whole has struggled with

racial and gender imbalances during the hiring process [169, 211] and other research critiques the field's focus on algorithmic solutions to larger societal problems [26]. While this thesis did not examine the role of adversarial attacks and defences in bias mitigation, other researchers have discussed this [40, 216, 210].

3.8.4 Deferred Expertise

It is not uncommon for users to trust modern ML systems more than they should [78, 62, 100, 134, 172]. There are no shortage of examples of drivers falling asleep while operating Tesla cars in "full self-driving" mode, sometimes leading to fatal driving accidents [78, 62, 100]. Lawyers have faced penalties for inappropriate usage of chatbots wherein case law was invented wholesale by models not designed to produce accurate text [134]. One study showed that individuals who use ML-based code assistance not only trusted their code more, but were also more likely to produce code with security flaws [162]. Another study showed that the accuracy of medical diagnoses is expected to decrease between 5–30% with the introduction of ML-based assistance for medical doctors due to false confirmation errors, where doctors falsely confirm a diagnosis using ML [172].

These issues of misplaced trust and over-reliance on machine learning systems underscore a broader point: ML models, while powerful, are not autonomous experts. Their perceived authority often exceeds their actual capability, introducing new risks and compounding existing ones [134, 162, 172]. However, the implications of widespread machine learning adoption extend beyond human trust and safety. Even when models make their users fully aware of their limitations, the models come with substantial environmental costs [199, 2, 75].

3.8.5 Environmental Concerns

The development, training, and deployment of modern machine learning models demand enormous computational and data resources, which in turn translate to high energy consumption [2], hardware waste [199], and broader sustainability challenges [68, 75]. Furthermore, data collection can be very costly [171], increases development time [114], and makes it harder to verify a product [221]. As machine learning systems continue to scale in complexity and deployment, their environmental impact has become a growing concern among researchers [2, 3], policymakers [75], and practitioners [121]. While the societal benefits of ML are allegedly significant, they come at a substantial ecological cost that is often under-reported and poorly accounted for in system evaluations.

These training runs can span days [182], weeks [80], or even months [61] across thousands of high-performance GPUs, consuming megawatt-hours of electricity and contributing significantly to carbon emissions [187]. For example, the training of a single large language model has been estimated to emit

as much carbon as five average American cars over their entire lifetimes [187]. As models become larger and more capable, their environmental footprint continues to grow unless mitigated by advances in hardware efficiency, algorithmic optimisation, or the adoption of green energy sources.

Beyond training and inference, environmental costs accrue throughout the ML lifecycle. Data collection is not only time-consuming and labour-intensive [171], but also energy-intensive——particularly when involving data-intensive processes like web scraping, sensor networks, or other real-time data pipelines [2]. Preprocessing and cleaning this data can further increase compute overhead, extending development cycles [114] and complicating verification procedures [221]. Moreover, the continual deployment and fine-tuning of models in production environments adds to long-term infrastructure demands, especially in cases where retraining is frequent or large models are frequently synchronised across a network.

Hardware production for ML also presents a growing sustainability issue. The demand for GPUs and specialised accelerators has led to increased extraction of rare earth metals and non-renewable materials [199]. This contributes to electronic waste and accelerates hardware obsolescence as newer, more powerful chips are released to support ever-larger models [56].

These challenges have prompted early efforts to promote sustainable machine learning. Some proposals include carbon-aware scheduling of training jobs [9], algorithmic innovations like sparse training and pruning [91], and better reporting standards that include metrics like energy usage and emissions alongside traditional performance scores [9].

In the context of this thesis, environmental considerations directly influence choices such as model architecture, dataset size, and training strategies. As such, all research was conducted on a single graphics card (Papers I and III) or on a single processor (Papers II, and V), with the exception of Paper IV, which focused specifically on power optimisation and adversarial robustness in a cloud environment. Additionally, understanding the resource constraints that organisations face when defending their models is critical when evaluating the practicality of proposed adversarial defences. Thus, environmental impact is not just an abstract concern but a practical constraint that shapes both the feasibility and ethical responsibility of machine learning research.

As machine learning continues to be deployed in safety-critical domains, it is vital that environmental sustainability becomes a core consideration—just as safety, security, privacy, and reliability are. Ignoring these costs risks creating systems that may be technically impressive but environmentally unsustainable at scale and further exacerbate global inequities around who benefits from the deployment of these models.

While Chapter 2 outlined the regulatory and safety challenges surrounding machine learning, and this chapter has explored the structural and ethical dimensions of model development, it is crucial to recognise that even well-intentioned, well-designed, and well-regulated systems can be actively subverted—— an increasingly urgent concern addressed in next chapter.

Chapter 4

Adversaries

In security research, it is commonplace to discuss a system's safety in terms of a *worst-case scenario* that assumes that attackers are maximally knowledgable, well-resourced, and highly motivated [163]. While these assumptions might seem extreme, for complex and opaque systems like ML models, this worst-case evaluation is useful for understanding weaknesses and flaws in ML pipelines and critical for highlighting edge-case failures that would otherwise be unforeseen. However, in the context of ML, those unforeseen edge cases *can* be simulated using the aforementioned adversarial attacks—where a malicious user induces unintended behaviour of an ML model.

4.1 Computer Security

Before examining adversarial attacks in the ML context, it is useful to discuss the more mature landscape of computer and data security more broadly. The canonical worst-case scenario is remote code execution, wherein a malicious user located elsewhere is able to induce their desired behaviour on a victim's machine [163]. However, before code is executed, the adversary must gain access, via an *intrusion attack*, which might involve circumventing a network control mechanism [194] or exploiting a vulnerability in an application [149]. One goal of an intrusion might be to *exfiltrate* data, moving it from the user's control to the attacker's control. In some cases, the attacker is not concerned with stealing data or executing code; instead attackers can flood the application with *spam* or otherwise participate in a distributed, denial of service (DDoS) campaign. These classical attack types also have parallels in ML systems. Just as traditional applications can be intruded upon or disrupted, ML models can be manipulated, subverted, or overloaded through analogous adversarial attacks. Papers II and V use popular malware detection datasets to demonstrate the ease of subverting ML models even when those models can detect classical security breaches with great accuracy.

4.2 Adversarial Attacks for Machine Learning

ML has proven to be useful in safety-critical applications like system intrusion detection [104], network anomaly detection [131], image recognition [197], aviation [128], policing [193], security checkpoints [10], and medical imaging [58]. However, many such attacks threaten the real-world operation of these systems and can be grouped into several categories [152] *inference* attacks can be used to glean information about the training data [123, 181, 44], *extraction* attacks reveal private information about the model [92], *poisoning* attacks change model behaviour by targeting the training data [178, 23, 22], and *evasion* attacks change the model behaviour by targeting the model at run-time [36, 37, 145, 42, 110, 43]. The work in this theses focuses on the last category, evasion attacks, because of the ability of that class of attacks to induce failures at run-time, thus allowing for the failure rate analyses discussed in Chapter 1.3.5 and 2. While much work has gone into evaluating the robustness for machine learning models [145, 24, 36, 54], these models exist in the real-world, despite the extreme ease of breaking them [139, 24, 36, 140].

4.2.1 Threat Model

In adversarial ML, a threat model characterises the attacker's knowledge, capabilities, and objectives. Understanding these aspects is essential for assessing system vulnerabilities and designing robust defences.

We begin by defining key terms related to attacker knowledge. In a *white-box* setting, the attacker is assumed to have full access to the model, including its architecture, parameters, and possibly the training data. This level of access enables the most powerful attacks, as the adversary can exploit detailed knowledge of the model's internals to craft highly effective attacks. In contrast, a *black-box* attacker is restricted to observing the model's outputs, such as class labels or confidence scores, without access to its internal structure or weights. Between these extremes, *gray-box* attackers may have partial information, such as knowledge of the model architecture or general characteristics of the training data.

The threat model also defines the attacker's goals, *e.g.*, targeted or untargeted attacks. Targeted attacks attempt to cause a specific behaviour, steering the model towards a particular incorrect output. Untargeted attacks aim to induce any incorrect prediction, regardless of the output target. Confidence reduction attacks seek to undermine the model's certainty without necessarily changing its top prediction, degrading overall trust in the system. There are many more classes of attacks, and those are outlined in the subsections below.

An important aspect of adversarial ML is that attacks can escalate a threat model from black-box assumptions toward white-box capabilities. Even if an adversary initially has limited access to a model, techniques such as model extraction, inference, and membership attacks can allow them to progressively increase their knowledge.

For example, Tramèr et al. [192] demonstrated that querying a ML API with $p + 1$ random p-dimensional inputs could reveal enough information to reconstruct a model's decision boundary, thereby enabling model extraction from mere prediction outputs. Similarly, Fredrikson et al. [63] showed that attackers can infer sensitive attributes or reverse-engineer category mappings using limited model outputs. Shokri et al. [181] proposed membership inference attacks that identify whether a particular sample was part of the training set by analysing model responses across multiple low-cost proxy models for the targeted original. Moreover, adversarial examples crafted against surrogate models often exhibit *transferability*, meaning they remain effective against the original target model even without direct access [220, 192].

These attacks, initially exploiting only hard labels or soft confidence scores, progressively reveal private information about the model or training set, transitioning a black-box attacker towards more informed, gray-box or white-box capabilities. Thus, in practice, the strict separation between black-box and white-box threat models is porous. Attackers may start with minimal assumptions but, through strategic queries and statistical analysis, gain detailed insight into the model, expanding their capabilities and increasing the severity of potential attacks. In short, ML does not make the training data or model parameters secure [5].

Finally, attacks can be categorised based on the phase of the ML pipeline they target. Training-time attacks such as poisoning introduce malicious data during model development. Inference-time attacks, such as evasion attacks, manipulate model inputs after deployment to induce misclassifications or large changes in a regression output. Post-deployment attacks, including model extraction and membership inference attacks, aim to recover sensitive information or replicate the model through external interactions alone. Given this fluid escalation of attacker knowledge and capabilities, it is crucial that threat models are explicitly stated and considered throughout the design, evaluation, and deployment of ML systems.

Furthermore, by designing systems that are robust to attacks, we can ensure models that are safer, more private, and more reliable across all use cases and diverse users and bystanders. Papers I and II develop this thinking. Papers III and IV formalise risk analysis for adversarial attacks. Paper V proposes a model that is safer-by-design and Paper VI presents a software framework for incorporating these evaluations into all ML pipelines.

4.2.2 Extraction Attacks

While the white-box threat model assumes the adversary has access to the model parameters, the model's decision boundary or model parameters can be approximated through statistical techniques or via other attacks [203, 92, 20]. In many cases, perfect knowledge of a modelling process is provided by scientific research publications and publicly available source code. However, most commercial models are proprietary and can only be accessed through

an API that returns only the classification (*e.g.*, as confidence score or the classification labels) [192, 92]. In either case, creating a functionally identical copy of a publicly available model can be substantially cheaper than training one— a large model can be copied (with 90% of the original performance) in only 100k queries [116]. In the case of continuous ML functions (*e.g.*, logistic regression), a straight-forward approximation method is known to quickly find counter-examples [71]. In the case of discontinuous ML functions (*e.g.*, decision trees), a continuous auxiliary function can still be used [192, 103]. In all cases, the decision boundary or regression function can be approximated by using relatively small models [18, 42].

4.2.3 Inference Attacks

Attacks that target private aspects of the training data are called inference attacks [123, 96, 181]. These attacks leverage access to a trained model's predictions to infer sensitive characteristics of the data it was trained on, such as membership, class distributions, or even specific attribute values. Inference attacks are of particular concern in regulated domains such as healthcare or law enforcement (or, really, any service that wishes to serve EU residents. See Chapter 2.3), where models trained on sensitive personal data may inadvertently leak private information.

One of the most well-known examples is the membership inference attack, where an adversary determines whether a particular data point was used in training the model [181, 123]. By exploiting differences in confidence scores or output behaviour between seen (training) and unseen (non-training) inputs, an attacker can classify whether a sample was part of the original training set. Other forms of inference attacks aim at more detailed information. For instance, attribute inference attacks aim to predict sensitive or hidden attributes of training samples, given partial access to input features and the corresponding model outputs [63, 11]. These attacks are especially potent in healthcare, recommendation systems, and finance, where partial demographic or behavioural features may be accessible, and the goal is to uncover private attributes like medical conditions or credit status. More recently, label-only inference attacks have emerged, where adversaries do not have access to confidence scores or model internals, but rely solely on the predicted class labels [44]. These black-box attacks use repeated querying and statistical analysis to infer sensitive properties of training data, even when access to output probabilities is restricted.

The severity of such attacks depends on model architecture, training procedures, and deployment conditions, making it imperative to assess and mitigate risks during model development and deployment.

4.2.4 Poisoning Attacks

Poisoning attacks target the model during its training phase by introducing carefully crafted malicious inputs into the training dataset. The goal of poisoning is to subvert the learning process so that the trained model exhibits attacker-controlled behaviours, such as misclassifying specific inputs, degrading overall performance, or embedding hidden backdoors that can be used by an attacker to trigger specific behaviour at run-time.

There are two primary types of poisoning attacks: *availability attacks* and *integrity attacks* [97]. Availability attacks aim to cause general model failure, degrading performance across a broad range of inputs and rendering the system unusable or unreliable [97]. These attacks are particularly concerning for safety-critical domains like autonomous driving or medical diagnosis, where general failure can have catastrophic consequences. Integrity attacks, by contrast, are more subtle: they seek to degrade model performance on specific instances or tasks while maintaining overall accuracy elsewhere, thereby evading detection. A well-known example is backdoor or Trojan attacks, where an attacker introduces a specific pattern into the training data that, when present at inference time, causes misclassification into an attacker-chosen target class [176, 196, 180, 170, 184, 41].

A key challenge for poisoning attacks is stealth. If poisoned examples differ too much from clean examples, simple outlier detection methods [160] or robust training procedures might eliminate them [45]. Successful poisoning attacks therefore craft inputs that are both adversarial and statistically similar to genuine data.

Beyond individual samples, poisoning can also operate at the feature or label level—feature poisoning modifies the input features of training data without altering labels, nudging the decision boundary [212]; attacks that submit mislabelled data to reduce accuracy [95]; and attacks that scale efficiently against large models [67]. The threat of poisoning is exacerbated in distributed learning settings, such as federated learning, where training data is aggregated from many sources. In these scenarios, malicious clients can upload poisoned updates without direct oversight, creating significant vulnerabilities [13].

Given the serious risks posed by poisoning attacks, research into robust training procedures—such as differential privacy, robust optimisation, and certified defences—has grown rapidly in recent years [186, 205]. However, achieving strong resilience against poisoning attacks without sacrificing model performance remains an open problem.

4.2.5 Evasion Attacks

Evasion attacks aim to manipulate input data during the inference phase to deceive the model into misclassifying certain inputs or otherwise change the behaviour of a model [36, 31, 110, 42]. Evasion attacks vary largely by the optimisation criteria used to generate the adversarial examples.

Some of the most prominent techniques involve crafting small, often imperceptible perturbations to the input data that cause the model to make incorrect predictions while leaving the data visually or semantically unchanged to a human observer. Methods like the Fast Gradient Method (FGM) [71] and Projected Gradient Descent (PGD) [127] craft perturbations by following the gradient of the loss function with respect to the input data, maximising the model's prediction error. Other attacks, like DeepFool [145], seek minimal perturbations that move an input across the decision boundary.

Beyond small perturbations, more conspicuous forms of evasion have also been studied, such as adversarial patches [31], where localised and visible patterns placed on an object or image can reliably cause misclassification, regardless of the image they are applied to.

The feasibility of evasion attacks across a range of domains, including computer vision, cybersecurity, and natural language processing, highlights the pressing need for robust evaluation frameworks and defence mechanisms, such as adversarial training [127] and certified defences [73].

Chapter 5

Verification and Validation

The previous chapters introduced the ML pipeline, outlined the regulations that govern safety and privacy, examined some particular ML models, and described a very real threat model against them in the form of attacks. In contrast, this chapter discusses the ways in which this thesis adheres to and promotes best practices for reproducibility and auditability. Verification and validation (V&V) broadly refers to the processes used to ensure that a system meets its requirements and behaves safely and reliably under expected– and unexpected– conditions. V&V methods can broadly be classified into several categories: formal verification, simulation, empirical modelling, and static or dynamic analysis. Fields like aviation, automotive, and medicine use some combination of these to mitigate and control risks on a daily basis.

5.1 Formal Verification

Formal verification is a method of mathematically proving the correctness of a system with respect to a formal specification. Unlike simulation, which tests a system under selected scenarios, formal verification attempts to exhaustively prove that certain properties (*e.g.*, safety, robustness, or correctness) always hold, regardless of the input or system state. This makes it especially attractive in systems where failure is catastrophic and exhaustive testing is infeasible.

The strength of formal verification lies in its guarantees: if a property is proven, it holds for all cases within the bounds of the assumptions made. However, this rigour comes with significant costs and complexity. Creating formal specifications requires a precise and often abstract understanding of the system, and the proofs themselves can be highly non-trivial, requiring expert knowledge and substantial computational resources [133, 64].

An important example of formal verification is the CompCert C compiler, which has been formally verified to produce assembly code that is functionally equivalent to its source code [120]. In contrast to conventional compilers, which

may contain bugs in optimisation or code generation, CompCert offers mathematical guarantees of correctness, making it appealing for critical systems such as avionics and medical devices. In medical devices, formal verification has been applied to implantable cardiac pacemakers, which must maintain life-critical timing constraints to not accidentally kill the person they are meant to help. Research efforts, such as those of Jiang et al. [98], have demonstrated how formal models of pacemaker logic can be verified to ensure that the device will never enter dangerous states, even under extreme timing conditions or unexpected cardiac rhythms.

In the domain of machine learning, formal verification is becoming increasingly relevant, as ML models are being deployed in safety-critical applications such as autonomous vehicles, medical diagnostics, and security systems. Verifying the correctness of ML models with respect to a formal specification presents unique challenges compared to traditional software systems. Unlike conventional software, ML models are typically data-driven and may exhibit unpredictable behaviour due to the complex nature of their learned representations. Despite these challenges, researchers have developed techniques for formal verification of certain aspects of ML models, such as ensuring robustness to adversarial attacks or proving the correctness of a model's decision-making process in specific environments. While the field is still in its infancy, work has been done in verifying neural networks' robustness to adversarial perturbations and ensuring that certain fairness constraints are met during the learning process [133]. However, these formal methods can be computationally infeasible or altogether useless during model selection [86].

As ML models continue to play a larger role in critical applications, the need for formal verification to guarantee their reliability and safety will only grow. While formal verification guarantees the correctness of a system through mathematical proofs, simulation provides a way to find how common error states and other extreme conditions are.

5.2 Simulation

Simulation is a powerful verification technique in which complex systems are modelled and tested under a variety of conditions before real-world deployment. These techniques are especially prominent in high-stakes domains such as aerospace, automotive safety, and medical science, where real-world testing is expensive, dangerous, or ethically challenging.

In the aerospace industry, simulation is integral to the development and certification of an aircraft. One well-known example is the Boeing 777, which was the first commercial aircraft to be entirely designed using computer-aided design (CAD) and simulation tools, minimising the need for physical prototypes [207]. It is also famous for being the first "fly-by-wire" system where manual pilot input was replaced with electrical signals and onboard computers which forced the industry to simulate physical interactions in software to

ensure safety [207].

In automotive engineering, crash simulations are widely used to complement physical crash testing. However, these simulations often rely on models based on the median male body, leading to safety designs that disproportionately protect average-sised men. Research has shown that women are significantly more likely to suffer injury in car crashes due to this bias in crash test dummies and the resulting simulations [174].

Similarly, in medical science, simulations of drug interactions or disease progression are often built from datasets that over-represent male physiology [135]. This has led to a "male body problem" in medical research, where treatments may be less effective or riskier for women. Additionally, statistical analyses in clinical trials often rely heavily on p-values, which can misrepresent the reliability of results when not properly contextualised—known as the p-value problem [72].

Unlike physical systems that are governed by well-known physical laws, many ML models can be hard to explain or interpret [124]. By harnessing adversarial attacks to simulate worst-case samples, model builders can test the robustness and reliability of machine learning systems under extreme or intentionally challenging conditions. Adversarial simulation involves creating perturbed inputs that are specifically designed to cause model failures, revealing vulnerabilities that may not appear under standard testing conditions [71]. This approach enables practitioners to stress-test models systematically before they are deployed in the real-world. However, there is a data-driven approach that relies on real-world experimentation call.

5.3 Empirical Modelling

Empirical models can take various forms, such as statistical models, ML models (Chapter 3), or system dynamics models (*e.g.* structural or mechanical engineering [82]). For example, a simple statistical model is the exponential distribution, which is often applied in reliability engineering to model the time between events in systems that experience a constant failure rate over time. A classic example from reliability engineering is the exponential distribution, which models the time between failures in systems that exhibit a constant failure rate. This distribution is defined by a single parameter, λ, the rate of failure (or the inverse of the mean time between failures). Its probability density function (PDF) is:

$$f(t) = \lambda e^{-\lambda t} \quad \text{for } t \geq 0,$$

where $f(t)$ denotes the likelihood of failure at time t. While analytically simple, this model assumes a constant hazard rate, which is often unrealistic for complex systems.

More flexible are survival models, which do not assume a constant failure rate and are therefore better suited for modelling the complex relationship

between times, cost, and robustness. These models are explored in detail in Paper III, where survival time is used as a proxy for model robustness, and in Paper IV, which employs accelerated failure time (AFT) models to analyse adversarial robustness across neural networks trained on different hardware architectures.

In an AFT model, the log of survival time is modelled as a linear function of covariates:

$$\log(T) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \varepsilon,$$

where T is the survival time, x_i are covariates (e.g., model depth, learning rate, hardware type), β_i are model coefficients, and ε is a random error term. This allows interpretation of covariates as multiplicative effects on the survival time, offering a more intuitive understanding of robustness under adversarial pressure.

Survival models play a critical role across a range of domains where understanding time-to-event behaviour is essential. In medicine, survival analysis is extensively used in epidemiological studies to model patient survival times, time until disease recurrence, or time to recovery [30]. In medicine, covariates such as age, treatment type, genetic markers, and co-morbidities are incorporated to estimate patient-specific risks and guide personalised treatment strategies [30]. In the automotive domain, survival models help assess the reliability and durability of components, such as engines, brake systems, and electrical subsystems. By analysing failure times under different usage conditions and environments, manufacturers can improve design, schedule preventive maintenance, and optimise warranty policies [112]. In aircraft design, survival analysis informs the estimation of failure probabilities for critical components under stress, load, and fatigue over time [206]. These models support decisions in structural engineering, maintenance scheduling, and certification by regulatory authorities to ensure high standards of safety and reliability throughout the aircraft's operational lifecycle [206].

5.4 Static and Dynamic Analysis

When building a bridge, static analyses are used to estimate the forces acting on a component when a part is not moving and dynamic analyses are used to estimate forces when acceleration is experienced [82]. This same metaphor can be applied to software development—where static analysis examines code for errors, vulnerabilities, and compliance issues without executing it [19], while dynamic analysis tests and monitors the software during runtime to uncover defects that only appear during execution [15]. These two approaches were applied throughout this thesis to ensure the correctness, reliability, and reproducibility of the developed software and experimental results.

In automotive or aviation, physical components such as engines, control surfaces, or braking systems are subjected to both static and dynamic analyses

to verify their integrity under both constant and changing loads. Similarly, embedded software controlling these components undergoes static analysis to catch coding errors early and dynamic analysis to validate behaviour under real-world operational conditions [207]. In medical or banking software, static analysis is critical to ensure compliance with regulatory standards like HIPAA or GDPR by identifying security vulnerabilities and logical errors before deployment, while dynamic analysis is employed to validate the correctness, reliability, and security of live systems handling sensitive patient or financial data [94, 113].

Throughout this thesis, static analyses in the form of pre-commit scripts [183] and unit tests [161] were used to ensure code quality, consistency, and the early detection of potential errors before code was used for experimentation as well as in the generation of the included papers. Pre-commit hooks are automated scripts that run a set of checks—such as linting, formatting enforcement, or static code analysis—each time a developer attempts to make a change to the source code, effectively acting as a gatekeeper to prevent problematic code from entering the codebase [183]. All papers were formatted for consistency in this way and all source code was evaluated for things like unreachable code, undeclared variables, and un-tested functions. In addition, functional and integration tests were used to conduct dynamic analyses of the software components developed. Functional tests verify that individual functions or features perform as expected when executed and integration tests, on the other hand, validate that different modules or services interact correctly and robustly when combined, ensuring that defects that only emerge from complex system interactions are detected and addressed during runtime [201]. Furthermore, each paper and the source code for the experiments has been made available as part of the software package described in Paper VI thus ensuring that anyone can verify and validate the findings presented in this thesis.

5.5 Accuracy is Not Enough

As we move from modelling individual components to reasoning about system-level reliability, a more fundamental question arises: "how can we trust complex systems whose behaviours are shaped by large-scale, opaque, and data-dependent models?"

Historically, marginal gains in model performance have come at the cost of exponentially larger models [56]. Large models require increasingly vast datasets [56, 200, 28], which are collected, catalogued, distributed by narrowing number of elite and well-funded institutions [107]. By amplifying systematic biases, this concentration introduces critical risks, including gender bias [126], racial discrimination [34], and fatal design flaws [16]. These concerns are not new [48, 168, 34]; they echo through decades of evidence showing higher fatality rates for female-bodied individuals in car accidents [60], or neural networks that inadvertently encode racial information from medical images [69].

Beyond social and ethical implications, data collection and labelling for machine learning itself introduces a host of practical concerns. It is expensive [171], raises significant privacy issues [27], extends time-to-market [114], slows innovation [221], and relies on vast numbers of outsourced workers [25]. Worse still, performance metrics in machine learning often paint an overly optimistic picture [127, 51]. As researchers have noted [127, 36, 50, 138], traditional test/train evaluation only uncovers failures within the training distribution which leaves models vulnerable to catastrophic failures when deployed in the real world.

To build truly trustworthy systems, model builders must go beyond conventional benchmarks and provide strong, actionable guarantees. These guarantees must address the practical, social, and technical limitations outlined above——and they must do so efficiently. In particular, we need rigorous, cost-effective methods for verifying model behaviour under diverse and challenging conditions, ideally fast enough to be integrated into development feedback loops and used in simulation, rather than relying on deployment in real-world systems like autonomous vehicles, delivery drones, or manufacturing robots.

Papers I and II examines the run-time and efficacy of several adversarial attacks and model defences in an effort to find which defences are trustworthy. Paper III presents a metric for quantifying the trust of a model against a given set of attacks and Paper IV uses this to build models that are not only more trustworthy, but also more power and cost efficient. Paper V then outlines a model that aims to keep user data private and thus be more trustworthy by default. Finally, Paper VI outlines a safety-critical framework for ML verification and validation and presents the software tooling required to not only reproduce the work included in this thesis, but to conduct the same type of massive and rigorous evaluations of any ML model trained on any dataset.

Chapter 6

Summary of Contributions and Future Work

The papers below first quantify the problem of adversarial attacks, explore the efficacy and cost of different defences, and presents a novel framework for approaching and certifying the trustworthiness of a system.

6.1 Paper I

Safety-critical computer vision: an empirical survey of adversarial evasion attacks and defenses on computer vision systems. Charles Meyers, Tommy Löfstedt, and Erik Elmroth. Artificial Intelligence Review, Volume 56, 2023.

In this paper, five years of attacks and defences were surveyed and a novel run-time analysis was included. In this way, the marginal cost and benefit for a given defence with respect to the performance on perturbed (adversarial) and unperturbed data was quantified. This paper first noted the adversarial failure rate as an upper-bound to the real-world failure rate and concluded that no known model is safe according to international standards.

6.2 Paper II

Massively parallel evasion attacks and the pitfalls of adversarial retraining. Charles Meyers, Tommy Löfstedt, and Erik Elmroth. EAI Endorsed Transactions on Internet of Things, Volume 10, 2024.

This paper shows that, in practice, an attacker can find adversarial examples in linear time, giving them a distinct advantage over polynomial-time model builders. The effect of noise distance, model run-time, and adversarial retraining on adversarial robustness was examined across several anomaly detection

benchmark datasets. This paper raises questions about the cost of robustness in terms of both compute time and benign accuracy.

6.3 Paper III

Training Rate and Survival Heuristic for Inference and Robustness Evaluation (Trashfire). Charles Meyers, Mohammad Reza Saleh Sedghpour, Tommy Löfstedt, and Erik Elmroth. IEEE International Conference on Machine Learning and Cybernetics (ICMLC), Volume 1, 2024.

This paper formalises a survival analysis method for machine learning applications that quantifies model robustness in terms of survival time. By comparing this robustness measure to training time, a metric is introduced to quickly reject model architectures where the attacker has the advantage. This ratio is presented and dubbed the training rate and survival heuristic, or TRASH score. The effect of training epochs and model depth on the ResNet model was examined across several benchmark image datasets to show that deeper models are not more robust.

6.4 Paper IV

A Cost-Aware Approach to Adversarial Robustness in Neural Networks. Charles Meyers, Mohammad Reza Saleh Sedghpour, Erik Elmroth, and Tommy Löfstedt. Submitted for publication September 2024.

Building on the previously developed tooling, the training rate and survival heuristic from Paper III was applied to model the robustness of machine learning models as a function of various parameters. Rather than focusing on model depth and training epochs as in Paper III, this study investigated the impact of learning rate optimisation and hardware architecture on model robustness, using the survival models and parallel experimentation software developed earlier. A cost-aware, hardware-agnostic approach to model tuning was verified, demonstrating that models costing several dollars to train can be compromised for mere pennies. Additionally, low-cost 8-bit GPUs were shown to outperform commonly used 32-bit GPUs—not only in terms of cost-efficiency but also in adversarial robustness. The survival analysis technique from Paper III was further validated by evaluating the model in this new context.

6.5 Paper V

A Tiny, Client-Side Classifier. Charles Meyers, Aaron P. MacSween, Erik Elmroth, and Tommy Löfstedt. Manuscript, Umeå University, Sweden, 2024.

Prior research has consistently shown that more accurate models tend to be less robust, that subverting arbitrary model architectures is often trivial, and

that models are, at best, compressed representations of data intended to remain private. I n response to these limitations, a lightweight classifier was developed that performs well across diverse data types and with very small sample sizes. Additionally, the notion of compression distance was extended to kernel methods, resulting in improved run-time performance compared to state-of-the-art approaches. In addition, we offer improvements in run-time compared to the state-of-the-art.

6.6 Paper VI

Deckard: A tool for robust, declarative, and reproducible AI. Charles Meyers. Manuscript, Umeå University, Sweden, 2025.

To meet the legal requirements mentioned throughout the other works, it is necessary to build reproducible, audit-able, and explainable software. Due to the large number of hyper-parameters, specialised hardware requirements, huge amounts of data, and long run-times associated with modern ML models (*e.g.*, neural networks), managing and tracking a large number of experiments can be a headache. Furthermore, as robustness is discussed, it must always be framed in the context of a particular attack (or a set of attacks) since the dataset, model, and attack surface will vary in practice. While there are software solutions around optimising a model within a certain framework, they are not natively built for adversarial analysis.

Instead of committing to a particular framework (*e.g.*, Keras, Pytorch, or Tensorflow) and implementing new attacks from scratch in the chosen framework, one can use the proposed flexible software framework for expressing entire ML pipelines as declarative and human-readable files, which has been named `deckard`. Additionally, `deckard` allows one to run experiments across heterogeneous and distributed hardware and allows the experimenter to divide the pipeline arbitrarily across the available resources. By building a way to reliably reproduce ML experiments on arbitrary hardware platforms with arbitrary entry-points, using arbitrary software functions, rather than building around a particular ML framework, the software remains flexible and useful for a wide variety of existing tool-chains.

Finally, to account for traditional metrics and various robustness criteria, the software can be used with a variety of multi-objective optimisation algorithms, attack classes, and scoring functions. In this paper, the software is outlined as a step towards explainable and robust AI.

6.7 Future Work

This thesis contributes to the nascent field of trustworthy machine learning by introducing empirical techniques for quantifying model robustness, proposing a privacy-preserving classification model, and providing a platform for reproducible experimentation. Building on this foundation of safety and privacy, fairness and equity remain obvious next steps. One promising (but otherwise unexplored) direction is the detection and mitigation of demographic bias in both training and inference stages of ML systems. Time-to-failure (as outlined in Papers III & IV) can be used to find under-represented subsets in a test set, and poisoning attacks could be used to generate synthetic data and mitigate the effects of this under-representation. By shifting from training-time interventions like poisoning attacks to run-time interventions like evasion, extraction, or inference attacks, this approach could potentially detect and mitigate sampling bias, sensor failure, or any other scenario that can be simulated at run-time.

A second important direction is the development of certified robustness bounds that are grounded not only in adversarial threat models but also in fairness, privacy, and environmental metrics. Current robustness certification schemes often assume perfect knowledge of the model, its training procedure, and its deployment context—assumptions that are rarely met in practice. Future research could investigate certification frameworks that are probabilistic, scenario-dependent, or dynamically updated as systems interact with real-world environments. Papers III and IV provide examples of this data-driven approach to model certification.

Third, the work begun in `deckard` could be extended into a compliance layer for ML systems, where model deployments are automatically benchmarked against international standards for safety-critical software. Embedding adversarial robustness, fairness diagnostics, and energy efficiency directly into the ML development pipeline could help bridge the gap between academic research and industrial and regulatory needs.

Additionally, environmental sustainability must move from a passive reporting practice to an active optimisation target. This thesis outlined preliminary considerations for model cost-awareness, but future systems should incorporate carbon footprint, hardware lifespan, and resource consumption as first-class optimisation objectives alongside traditional metrics like accuracy and throughput. Techniques like carbon-aware scheduling, model quantisation, and lifetime-aware hardware selection present ripe opportunities for empirical study and real-world deployment.

Finally, a broader philosophical shift is required. While this thesis falls short of solving the problem of adversaries in ML systems, it is a step toward the deployment of ML systems that are not just accurate, but also responsible, private, reliable, secure, safe, and, above all, trustworthy. Future work must explicitly reject the culture of chasing benchmark accuracy scores without regard for ethical, societal, and ecological impact. To put it bluntly, at the time of publication, no truly trustworthy ML systems exist and it is therefore

incumbent on this author, and the community at large, to make that widely known and to continue on the path to making such systems a reality.

Ultimately, trustworthiness must become a measurable, actionable, and legally enforceable property of machine learning systems, rather than a vague aspiration or footnote in a publication next to the funding statement. This thesis has laid part of the groundwork for that future, but much remains to be built.

Bibliography

[1] Jan. 2022. URL: https://www.nist.gov/about-nist.

[2] In: *Goldman Sachs* (May 2024). URL: https://www.goldmansachs.com/insights/articles/AI-poised-to-drive-160-increase-in-power-demand.

[3] In: *Goldman Sachs* (May 2024). URL: https://www.goldmansachs.com/insights/articles/AI-poised-to-drive-160-increase-in-power-demand.

[4] ISO and IEC. *Information security, cybersecurity and privacy protection — Information security management systems — Requirements*. Standard. Geneva, CH: International Organization for Standardization, Mar. 2022.

[5] Hal Abelson et al. *Bugs in our Pockets: The Risks of Client-Side Scanning*. 2021. arXiv: 2110.07450 [cs.CR].

[6] Nur Ahmed, Muntasir Wahed, and Neil C Thompson. "The growing influence of industry in AI research". In: *Science* 379.6635 (2023), pp. 884–886.

[7] Sabeen Ahmed et al. "Failure detection in deep neural networks for medical imaging". In: *Frontiers in Medical Technology* 4 (2022), p. 919046.

[8] Tiago Almeida and Jos Hidalgo. *SMS Spam Collection*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5CC84. 2011.

[9] Lasse F Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. "Carbontracker: Tracking and predicting the carbon footprint of training deep learning models". In: *arXiv preprint arXiv:2007.03051* (2020).

[10] Daniel Arp et al. "Dos and don'ts of machine learning in computer security". In: *31st USENIX Security Symposium (USENIX Security 22)*. 2022, pp. 3971–3988.

[11] Giuseppe Ateniese et al. "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers". In: *International Journal of Security and Networks* 10.3 (2015), pp. 137–150.

[12] Saeideh Ghanbari Azar et al. “From Promise to Practice: A Study of Common Pitfalls Behind the Generalization Gap in Machine Learning”. In: *Transactions on Machine Learning Research* ().

[13] Eugene Bagdasaryan et al. “How to backdoor federated learning”. In: *International conference on artificial intelligence and statistics*. PMLR. 2020, pp. 2938–2948.

[14] Pierre Baldi et al. “Parameterized neural networks for high-energy physics”. In: *The European Physical Journal C* 76.5 (2016), pp. 1–7.

[15] Thoms Ball. “The concept of dynamic analysis”. In: *ACM SIGSOFT Software Engineering Notes* 24.6 (1999), pp. 216–234.

[16] Victoria A Banks, Katherine L Plant, and Neville A Stanton. “Driver error or designer error: Using the Perceptual Cycle Model to explore the circumstances surrounding the fatal Tesla crash on 7th May 2016”. In: *Safety science* 108 (2018), pp. 278–285.

[17] David GT Barrett, Ari S Morcos, and Jakob H Macke. “Analyzing biological and artificial neural networks: challenges with opportunities for synergy?” In: *Current opinion in neurobiology* 55 (2019), pp. 55–64.

[18] Thomas Bekman et al. “Practical black box model inversion attacks against neural nets”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2021, pp. 39–54.

[19] Moritz Beller et al. “Analyzing the state of static analysis: A large-scale evaluation in open source software”. In: *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. Vol. 1. IEEE. 2016, pp. 470–481.

[20] Jason W Bentley et al. “Quantifying membership inference vulnerability via generalization gap and other model metrics”. In: *arXiv preprint arXiv:2009.05669* (2020).

[21] Simone Bianco et al. “Benchmark analysis of representative deep neural network architectures”. In: *IEEE access* 6 (2018), pp. 64270–64277.

[22] Battista Biggio, Blaine Nelson, and Pavel Laskov. “Poisoning Attacks against Support Vector Machines”. In: *arXiv:1206.6389 [cs, stat]* (Mar. 2013). (Visited on 11/02/2020).

[23] Battista Biggio, Blaine Nelson, and Pavel Laskov. “Poisoning attacks against support vector machines”. In: *International Conference on Machine Learning* (2012).

[24] Battista Biggio et al. “Evasion attacks against machine learning at test time”. In: (2013), pp. 387–402.

[25] Tuba Bircan and Mustafa F Özbilgin. “Unmasking inequalities of the code: Disentangling the nexus of AI and inequality”. In: *Technological Forecasting and Social Change* 211 (2025), p. 123925.

[26] Abeba Birhane. “Algorithmic injustice: a relational ethics approach”. In: *Patterns* 2.2 (2021).

[27] Cara Bloom et al. “Self-driving cars and data collection: Privacy perceptions of networked autonomous vehicles”. In: *Symposium on Usable Privacy and Security (SOUPS)*. 2017.

[28] Anselm Blumer et al. “Learnability and the Vapnik-Chervonenkis dimension”. In: *Journal of the ACM* 36.4 (1989), pp. 929–965.

[29] Mariusz Bojarski et al. “End to end learning for self-driving cars”. In: *arXiv preprint arXiv:1604.07316* (2016).

[30] Mike J Bradburn et al. “Survival analysis part II: multivariate data analysis–an introduction to concepts and methods”. In: *British journal of cancer* 89.3 (2003), pp. 431–436.

[31] Tom B Brown et al. “Adversarial patch”. In: *arXiv:1712.09665* (2017).

[32] Erik Brynjolfsson. “The turing trap: The promise & peril of human-like artificial intelligence”. In: *Daedalus* 151.2 (2022), pp. 272–287.

[33] Till J Bungert, Levin Kobelke, and Paul F Jaeger. “Understanding silent failures in medical image classification”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2023, pp. 400–410.

[34] Joy Buolamwini and Timnit Gebru. “Gender shades: Intersectional accuracy disparities in commercial gender classification”. In: *Conference on fairness, accountability and transparency*. PMLR. 2018, pp. 77–91.

[35] Daniel Caballero-Martin et al. “Artificial intelligence applied to drone control: A state of the art”. In: *Drones* 8.7 (2024), p. 296.

[36] Nicholas Carlini and David Wagner. “Towards evaluating the robustness of neural networks”. In: (2017), pp. 39–57.

[37] Nicholas Carlini et al. “On evaluating adversarial robustness”. In: *arXiv:1902.06705* (2019).

[38] Anirban Chakraborty et al. “Adversarial Attacks and Defences: A Survey”. In: *arXiv:1810.00069 [cs, stat]* (2018).

[39] Alan Chan et al. “The limits of global inclusion in AI development”. In: *arXiv preprint arXiv:2102.01265* (2021).

[40] Hongyan Chang et al. “On adversarial bias and the robustness of fair machine learning”. In: *arXiv preprint arXiv:2006.08669* (2020).

[41] Bryant Chen et al. “Detecting backdoor attacks on deep neural networks by activation clustering”. In: *arXiv preprint arXiv:1811.03728* (2018).

[42] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. “HopSkipJumpAttack: A query-efficient decision-based attack”. In: *IEEE symposium on security and privacy (sp)*. IEEE. 2020, pp. 1277–1294.

[43] Pin-Yu Chen et al. “Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models”. In: *Proceedings of the 10th ACM workshop on artificial intelligence and security*. 2017, pp. 15–26.

[44] Christopher A Choquette-Choo et al. “Label-only membership inference attacks”. In: *International conference on machine learning*. PMLR. 2021, pp. 1964–1974.

[45] Celia Cintas et al. “Detecting Adversarial Attacks via Subset Scanning of Autoencoder Activations and Reconstruction Error”. en. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. Yokohama, Japan, July 2020, pp. 876–882. ISBN: 978-0-9992411-6-5. DOI: 10.24963/ijcai.2020/122. URL: https://www.ijcai.org/proceedings/2020/122 (visited on 11/03/2020).

[46] Olga Ciobanu-Caraus et al. “A critical moment in machine learning in medicine: on reproducible and interpretable learning”. In: *Acta Neurochirurgica* 166.1 (2024), p. 14.

[47] Sam Corbett-Davies and Sharad Goel. “The measure and mismeasure of fairness: A critical review of fair machine learning”. In: *arXiv preprint arXiv:1808.00023* (2018).

[48] William A Corsaro. “Something old and something new: The importance of prior ethnography in the collection and analysis of audiovisual data”. In: *Sociological Methods & Research* 11.2 (1982), pp. 145–166.

[49] Kimberlé Crenshaw. “Mapping the margins: Identity politics, intersectionality, and violence against women”. In: *Stanford Law Review* 43.6 (1991), pp. 1241–1299.

[50] Francesco Croce and Matthias Hein. “Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks”. In: (2020), pp. 2206–2216.

[51] Francesco Croce and Matthias Hein. “Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks”. In: *International Conference on Machine Learning* (2020).

[52] Francesco Croce et al. “RobustBench: a standardized adversarial robustness benchmark”. In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. 2021. URL: https://openreview.net/forum?id=SSKZPJCt7B.

[53] Catherine D’Ignazio and Lauren F. Klein. *Data feminism*. The MIT Press, 2023.

[54] Pratyush Kr Deka et al. “Adversarial Impact on Anomaly Detection in Cloud Datacenters”. In: *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE. 2019, pp. 188–18809.

[55] Li Deng. “The mnist database of handwritten digit images for machine learning research”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[56] Radosvet Desislavov, Fernando Martínez-Plumed, and José Hernández-Orallo. “Compute and energy consumption trends in deep learning inference”. In: (2021).

[57] Elvis Dohmatob. “Generalized No Free Lunch Theorem for Adversarial Robustness”. In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. PMLR. 2019.

[58] Bradley J Erickson et al. “Machine learning for medical imaging”. In: *Radiographics* 37.2 (2017), pp. 505–515.

[59] European Commission. “Proposal for a Regulation of the European Parliament and of the Council laying down rules to prevent and combat child sexual abuse”. In: (May 2022). URL: `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM:2022:209:FIN`.

[60] Leonard Evans and Peter H Gerrish. “Gender and age influence on fatality risk from the same physical impact determined using two-car crashes”. In: *SAE transactions* (2001), pp. 1336–1341.

[61] Facebookresearch. *Facebookresearch/llama: Inference code for Llama Models*. URL: `https://github.com/facebookresearch/llama`.

[62] Tim Fitzsimons. *Sleeping tesla driver caught on Swedish Highway – after 25 miles*. May 2021. URL: `Tesla%20driver%20slept%20as%20car%20was%20going%20over%2080%20mph%20on%20Autopilot,%20Wisconsin%20officials%20say`.

[63] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. “Model inversion attacks that exploit confidence information and basic countermeasures”. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015, pp. 1322–1333.

[64] Daniel J Fremont et al. “Formal scenario-based testing of autonomous vehicles: From simulation to the real world”. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2020, pp. 1–8.

[65] Matjaz Gams and Sebastjan Kramar. “Evaluating ChatGPT’s consciousness and its capability to pass the Turing test: A comprehensive analysis”. In: *Journal of Computer and Communications* 12.3 (2024), pp. 219–237.

[66] Ji Gao et al. “Deepcloak: Masking deep neural network models for robustness against adversarial samples”. In: *arXiv preprint arXiv:1702.06763* (2017).

[67] Jonas Geiping et al. “Witches’ brew: Industrial scale data poisoning via gradient matching”. In: *arXiv preprint arXiv:2009.02276* (2020).

[68] A Shaji George, AS Hovan George, and AS Gabrio Martin. "The environmental impact of ai: A case study of water consumption by chat gpt". In: *Partners Universal International Innovation Journal* 1.2 (2023), pp. 97–104.

[69] Judy Wawira Gichoya et al. "AI recognition of patient race in medical imaging: a modelling study". In: *The Lancet Digital Health* 4.6 (2022), e406–e414.

[70] Bernardo Gonçalves. "The Turing test is a thought experiment". In: *Minds and Machines* 33.1 (2023), pp. 1–31.

[71] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv:1412.6572* (2014).

[72] Steven N Goodman. "P values, hypothesis tests, and likelihood: implications for epidemiology of a neglected historical debate". In: *American Journal of Epidemiology* 137.5 (1993), pp. 485–496.

[73] Sven Gowal et al. "On the effectiveness of interval bound propagation for training verifiably robust models". In: *arXiv preprint arXiv:1810.12715* (2018).

[74] Nico Grant and Kashmir Hill. *Google's photo app still can't find gorillas. and neither can Apple's.* May 2023. URL: https://www.nytimes.com/2023/05/22/technology/ai-photo-labels-google-apple.html.

[75] Michael Greshko. "Nuclear power for AI: what it will take to reopen Three Mile Island safely". In: *Nature* (Sept. 2024). DOI: 10.1038/d41586-024-03162-2. URL: https://www.nature.com/articles/d41586-024-03162-2.

[76] Odd Erik Gundersen and Sigbjørn Kjensmo. "State of the art: Reproducibility in artificial intelligence". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[77] Odd Erik Gundersen, Saeid Shamsaliei, and Richard Juul Isdahl. "Do machine learning platforms provide out-of-the-box reproducibility?" In: *Future Generation Computer Systems* 126 (2022), pp. 34–47.

[78] Simon Hamelius. *Sleeping tesla driver caught on Swedish Highway – after 25 miles*. Apr. 2024. URL: https://www.vibilagare.se/english/sleeping-tesla-driver-caught-swedish-highway-after-25-miles.

[79] Karen Hao. "The two-year fight to stop Amazon from selling face recognition to the police". In: *MIT Technology Review* (June 2020). Accessed: 2025-04-28. URL: https://www.technologyreview.com/2020/06/12/1003482/amazon-stopped-selling-police-face-recognition-fight/.

[80] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[81] Jose Hernandez-Orallo. “Beyond the Turing test”. In: *Journal of Logic, Language and Information* 9 (2000), pp. 447–466.

[82] R. C. Hibbeler. *Engineering mechanics. statics and Dynamics*. 14th ed. Pearson, 2016.

[83] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[84] Douglas R Hofstadter. *Gödel, Escher, Bach: an eternal golden braid*. Basic books, 1999.

[85] Bell Hooks. *Ain't I A woman: Black women and feminism*. Routledge, Taylor & Francis Group, 2015.

[86] WuLing Huang et al. “Autonomous vehicles testing methods review”. In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2016, pp. 163–168.

[87] Matthew Hutson. “Missing data hinder replication of arti2cial intelligence studies”. In: *Science (Feb. 15, 2018). https://doi. org/10.1126/science. aat3298* (2018).

[88] International Electrotechnical Commission. *IEC 61508 Safety and Functional Safety*. 2nd. International Electrotechnical Commission, 2010.

[89] International Electrotechnical Commission. *IEC 62304 Medical Device Software - Software Life Cycle Processes*. 2nd. International Electrotechnical Commission, 2006.

[90] International Standards Organization. *ISO 26262-1:2011, Road vehicles — Functional safety*. https://www.iso.org/standard/43464.html (visited 2022-04-20). 2018.

[91] Ashhadul Islam, SAMIR BELHAOUARI, and Amine Bermak. “Sustainable AI: Efficient Pruning of Large Language Models in Resource-Limited Environments”. In: *Workshop on Machine Learning and Compression, NeurIPS 2024*. 2024.

[92] Matthew Jagielski et al. “High accuracy and high fidelity extraction of neural networks”. In: *29th USENIX security symposium (USENIX Security 20)*. 2020, pp. 1345–1362.

[93] Daniel Jakubovitz and Raja Giryes. “Improving dnn robustness to adversarial attacks using jacobian regularization”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 514–529.

[94] Raoul Praful Jetley, Paul L Jones, and Paul Anderson. “Static analysis of medical device software using CodeSonar”. In: *Proceedings of the 2008 workshop on Static analysis*. 2008, pp. 22–29.

[95] Rishi Jha, Jonathan Hayase, and Sewoong Oh. “Label poisoning is all you need”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 71029–71052.

[96] Jinyuan Jia and Neil Zhenqiang Gong. "{AttriGuard}: A practical defense against attribute inference attacks via adversarial machine learning". In: *27th USENIX Security Symposium (USENIX Security 18)*. 2018, pp. 513–529.

[97] Wenbo Jiang et al. "A flexible poisoning attack against machine learning". In: *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE. 2019, pp. 1–6.

[98] Zhihao Jiang et al. "Modeling and verification of a dual chamber implantable pacemaker". In: *International conference on tools and algorithms for the construction and analysis of systems*. Springer. 2012, pp. 188–203.

[99] Roman Jurowetzki et al. "The Privatization of AI Research (-ers): Causes and Potential Consequences–From university-industry interaction to public research brain-drain?" In: *arXiv preprint arXiv:2102.01648* (2021).

[100] KABC. *Caught on video: Tesla driver apparently asleep at the wheel on California freeway*. Feb. 2023. URL: `https://abc7ny.com/tesla-asleep-at-wheel-sleeping-while-driving-driver-caught-snoozing/12777048/`.

[101] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. "Reinforcement learning: A survey". In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285.

[102] Maysa Khalil and Mohammad Azzeh. "Fake news detection models using the largest social media ground-truth dataset (TruthSeeker)". In: *International Journal of Speech Technology* (2024), pp. 1–16.

[103] Arnisha Khondaker and Nilanjan Ray. "Learning Instance-Specific Parameters of Black-Box Models Using Differentiable Surrogates". In: *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2025, pp. 7429–7438.

[104] Dong Seong Kim and Jong Sou Park. "Network-based intrusion detection with support vector machines". In: *International Conference on Information Networking*. Springer. 2003, pp. 747–756.

[105] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[106] Abigail MY Koay et al. "Machine learning in industrial control system (ICS) security: current landscape, opportunities and challenges". In: *Journal of Intelligent Information Systems* 60.2 (2023), pp. 377–405.

[107] Bernard Koch et al. "Reduced, reused and recycled: The life of a dataset in machine learning research". In: *arXiv preprint arXiv:2112.01716* (2021).

[108] Pang Wei Koh et al. "Wilds: A benchmark of in-the-wild distribution shifts". In: *International conference on machine learning*. PMLR. 2021, pp. 5637–5664.

[109] Jinia Konar, Prerit Khandelwal, and Rishabh Tripathi. “Comparison of various learning rate scheduling techniques on convolutional neural network”. In: *2020 IEEE International Students’ Conference on Electrical, Electronics and Computer Science (SCEECS)*. IEEE. 2020, pp. 1–5.

[110] Shashank Kotyan and Danilo Vasconcellos Vargas. “Adversarial robustness assessment: Why in evaluation both ℓ_0 and ℓ_∞ attacks are necessary”. In: *PloS one* 17.4 (2022), e0265723.

[111] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[112] Akhilesh Kumar, Ratna Babu Chinnam, and Alper Murat. “Hazard rate models for core return modeling in auto parts remanufacturing”. In: *International Journal of Production Economics* 183 (2017), pp. 354–361.

[113] Adam Prayogo Kuncoro, Imam Riadi, and Ahmad Luthfi. “Mobile forensics development of mobile banking application using static forensic”. In: *International Journal of Computer Applications* 975.1 (2017), p. 160.

[114] Hau Lam. “New design-to-test software strategies accelerate time-to-market”. In: *IEEE/CPMT/SEMI 29th International Electronics Manufacturing Technology Symposium (IEEE Cat. No. 04CH37585)*. IEEE. 2004, pp. 140–143.

[115] Richard J. Larsen and Morris L. Marx. *An introduction to mathematical statistics and its applications*. Pearson, 2018.

[116] Jhonatan Machado Leão et al. “Few-Shot Copycat: Improving Performance of Black-Box Attack with Random Natural Images and Few Examples of Problem Domain”. In: *International Conference on Pattern Recognition*. Springer. 2025, pp. 407–422.

[117] Mathias Lecuyer et al. “Certified Robustness to Adversarial Examples with Differential Privacy”. In: *2019 IEEE Symposium on Security and Privacy (SP)*. 2019, pp. 656–672.

[118] Legislature of the United States. *Children’s Online Privacy Protection Act*.
`https://www.govinfo.gov/content/pkg/USCODE-2011-title15/html/USCODE-2011-title15-chap91.htm`. 1998.

[119] Legislature of the United States. *Health Insurance Portability and Accountability Act*.
`https://www.govinfo.gov/content/pkg/PLAW-104publ191/pdf/PLAW-104publ191.pdf`. 1996.

[120] Xavier Leroy. “A formally verified compiler back-end”. In: *Journal of Automated Reasoning* 43 (2009), pp. 363–446.

[121] Baolin Li et al. “Clover: Toward sustainable ai with carbon-aware machine learning inference service”. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2023, pp. 1–15.

[122] Bo Li, Yevgeniy Vorobeychik, and Xinyun Chen. “A general retraining framework for scalable adversarial classification”. In: *Workshop on Adversarial Training, Neural Information Processing Systems* (2016).

[123] Zheng Li and Yang Zhang. “Membership leakage in label-only exposures”. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2021, pp. 880–895.

[124] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. “Explainable ai: A review of machine learning interpretability methods”. In: *Entropy* 23.1 (2020), p. 18.

[125] Dominic Loske and Matthias Klumpp. “Intelligent and efficient? An empirical analysis of human–AI collaboration for truck drivers in retail logistics”. In: *The International Journal of Logistics Management* 32.4 (2021), pp. 1356–1383.

[126] Kaiji Lu et al. “Gender bias in neural natural language processing”. In: *Logic, Language, and Security: Essays Dedicated to Andre Scedrov on the Occasion of His 65th Birthday* (2020), pp. 189–202.

[127] Aleksander Madry et al. “Towards deep learning models resistant to adversarial attacks”. In: *arXiv:1706.06083* (2017).

[128] Apoorv Maheshwari, Navindran Davendralingam, and Daniel A DeLaurentis. “A comparative study of machine learning techniques for aviation applications”. In: *2018 Aviation Technology, Integration, and Operations Conference*. 2018, p. 3980.

[129] Roger L McCarthy. “Autonomous vehicle accident data analysis: California OL 316 reports: 2015–2020”. In: *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering* 8.3 (2022), p. 034502.

[130] Edward McFowland, Skyler Speakman, and Daniel B Neill. “Fast generalized subset scan for anomalous pattern detection”. In: *The Journal of Machine Learning Research* 14.1 (2013), pp. 1533–1561.

[131] Tahir Mehmood and Helmi B Md Rais. “SVM for network anomaly detection using ACO feature subset”. In: *2015 International symposium on mathematical sciences and computing research (iSMSC)*. IEEE. 2015, pp. 121–126.

[132] Ninareh Mehrabi et al. “A survey on bias and fairness in machine learning”. In: *ACM computing surveys (CSUR)* 54.6 (2021), pp. 1–35.

[133] Mark Huasong Meng et al. “Adversarial robustness of deep neural networks: A survey from a formal verification perspective”. In: *IEEE Transactions on Dependable and Secure Computing* (2022).

[134] Sara Merken. *New York lawyers sanctioned for using fake CHATGPT cases in legal brief — reuters*. June 2023. URL: https://www.reuters.com/legal/new-york-lawyers-sanctioned-using-fake-chatgpt-cases-legal-brief-2023-06-22/.

[135] Lea Merone et al. "Sex inequalities in medical research: a systematic scoping review of the literature". In: *Women's Health Reports* 3.1 (2022), pp. 49–59.

[136] Domingo Mery et al. "Modern computer vision techniques for x-ray testing in baggage inspection". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47.4 (2016), pp. 682–692.

[137] Charles Meyers, Tommy Löfstedt, and Erik Elmroth. "Massively parallel evasion attacks and the pitfalls of adversarial retraining". In: *EAI Endorsed Transactions on Internet of Things* 10 (2024).

[138] Charles Meyers, Tommy Löfstedt, and Erik Elmroth. "Safety-critical computer vision: An empirical survey of adversarial evasion attacks and defenses on computer vision systems". In: *Springer Artificial Intelligence Review* (2023).

[139] Charles Meyers, Tommy Löfstedt, and Erik Elmroth. "Safety-critical computer vision: an empirical survey of adversarial evasion attacks and defenses on computer vision systems". In: *Artificial Intelligence Review* (2023), pp. 1–35.

[140] Charles Meyers et al. *A Training Rate and Survival Heuristic for Inference and Robustness Evaluation (TRASHFIRE)*. 2024. arXiv: 2401.13751 [cs.LG]. URL: https://arxiv.org/abs/2401.13751.

[141] Meyers et al. "A Systematic Approach to Robustness Modelling". In: *Springer Artificial Intelligence Review* (2023).

[142] Iman Mirzadeh et al. *GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models*. 2024. URL: https://arxiv.org/abs/2410.05229.

[143] Mana Moassefi et al. "Reproducibility of deep learning algorithms developed for medical imaging analysis: A systematic review". In: *Journal of Digital Imaging* 36.5 (2023), pp. 2306–2312.

[144] Yasir Abdelgadir Mohamed et al. "The impact of artificial intelligence on language translation: a review". In: *Ieee Access* 12 (2024), pp. 25553–25579.

[145] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2574–2582.

[146] Xiaomin Mou. "Artificial intelligence: investment trends and selected industry uses". In: *International Finance Corporation* 8.2 (2019), pp. 311–320.

[147] Mihai Nadin. “Intelligence at any price? A criterion for defining AI”. In: *AI & SOCIETY* 38.5 (2023), pp. 1813–1817.

[148] Mihai Nadin. “Intelligence at any price? A criterion for defining AI”. In: *AI & SOCIETY* 38.5 (2023), pp. 1813–1817.

[149] Mohammed Nasereddin et al. “A systematic review of detection and prevention techniques of SQL injection attacks”. In: *Information Security Journal: A Global Perspective* 32.4 (2023), pp. 252–265.

[150] Blaine Nelson et al. “Misleading learners: Co-opting your spam filter”. In: *Machine learning in cyber trust: Security, privacy, and reliability* (2009), pp. 17–51.

[151] Euclides Carlos Pinto Neto et al. “CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment”. In: *Sensors* 23.13 (2023), p. 5941.

[152] Maria-Irina Nicolae et al. “Adversarial Robustness Toolbox v1.2.0”. In: *CoRR* 1807.01069 (2018).

[153] Ryan van Nood and Christopher Yeomans. “Fairness as equal concession: critical remarks on fair AI”. In: *Science and Engineering Ethics* 27.6 (2021), p. 73.

[154] NSC. *Preliminary Semiannual Estimates of Motor Vehicle Fatalities*. 2025. URL: `https://injuryfacts.nsc.org/motor-vehicle/overview/preliminary-estimates/`.

[155] Luke Oakden-Rayner et al. “Hidden stratification causes clinically meaningful failures in machine learning for medical imaging”. In: *Proceedings of the ACM conference on health, inference, and learning*. 2020, pp. 151–159.

[156] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. “Knockoff nets: Stealing functionality of black-box models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4954–4963.

[157] World Health Organization. *Global status report on road safety 2023*. `https://www.who.int/teams/social-determinants-of-health/safety-and-mobility/global-status-report-on-road-safety-2023`. 2024.

[158] Nicolas Papernot et al. “Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks”. en. In: *arXiv:1511.04508 [cs, stat]* (Mar. 2016). URL: `http://arxiv.org/abs/1511.04508` (visited on 11/03/2020).

[159] Seymour A. Papert. *The summer vision project*. July 1966. URL: `https://dspace.mit.edu/handle/1721.1/6125`.

[160] Andrea Paudice et al. “Detection of adversarial training examples in poisoning attacks through anomaly detection”. In: *arXiv preprint arXiv:1802.03041* (2018).

[161] Jan Peleska. “Test automation for safety-critical systems: Industrial application and future developments”. In: *International Symposium of Formal Methods Europe*. Springer. 1996, pp. 39–59.

[162] Neil Perry et al. “Do users write more insecure code with AI assistants?” In: *Proceedings of the 2023 ACM SIGSAC conference on computer and communications security*. 2023, pp. 2785–2799.

[163] Inc. Petronella Technology Group. “NIST 800-53: Security and Privacy Controls for Information Systems and Organizations”. In: *National Institue of Standards* (Sept. 2020).

[164] George Philipp and Jaime G Carbonell. “Nonparametric neural networks”. In: *arXiv preprint arXiv:1712.05440* (2017).

[165] Eve Marie Phillips. “If it works, it’s not AI: a commercial look at artificial intelligence startups”. PhD thesis. Massachusetts Institute of Technology, 1999.

[166] Ayse Pinar Saygin, Ilyas Cicekli, and Varol Akman. “Turing test: 50 years later”. In: *Minds and machines* 10.4 (2000), pp. 463–518.

[167] Inioluwa Deborah Raji and Joy Buolamwini. “Actionable auditing: Investigating the impact of publicly naming biased performance results of commercial ai products”. In: *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. 2019, pp. 429–435.

[168] Deborah Ramirez, Jack McDevitt, and Amy Farrell. *A resource guide on racial profiling data collection systems: Promising practices and lessons learned*. US Department of Justice, 2000.

[169] Aneeta Rattan, Jennifer Steele, and Nalini Ambady. “Identical applicant but different outcomes: The impact of gender versus race salience in hiring”. In: *Group Processes & Intergroup Relations* 22.1 (2019), pp. 80–97.

[170] Ambrish Rawat, Killian Levacher, and Mathieu Sinn. “The devil is in the GAN: backdoor attacks and defenses in deep generative models”. In: *European Symposium on Research in Computer Security*. Springer. 2022, pp. 776–783.

[171] Yuji Roh, Geon Heo, and Steven Euijong Whang. “A survey on data collection for machine learning: a big data-ai integration perspective”. In: *IEEE Transactions on Knowledge and Data Engineering* 33.4 (2019), pp. 1328–1347.

[172] Rikard Rosenbacke et al. “AI and XAI second opinion: the danger of false confirmation in human–AI collaboration”. In: *Journal of Medical Ethics* (2024).

[173] Andrew Ross and Finale Doshi-Velez. "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 2018.

[174] Alyssa Ryan et al. "The impact of sex on motor vehicle crash injury outcomes". In: *Journal of Transportation Safety & Security* 14.5 (2022), pp. 818–842.

[175] Jeyabharathy Sadaiyandi et al. "Stratified sampling-based deep learning approach to increase prediction accuracy of unbalanced dataset". In: *Electronics* 12.21 (2023), p. 4423.

[176] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. "Hidden trigger backdoor attacks". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 07. 2020, pp. 11957–11965.

[177] Shibani Santurkar et al. "How does batch normalization help optimization?" In: *Advances in neural information processing systems* 31 (2018).

[178] Ali Shafahi et al. "Poison frogs! targeted clean-label poisoning attacks on neural networks". In: *Advances in neural information processing systems* 31 (2018).

[179] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, Cambridge, UK., 2014.

[180] Reza Shokri et al. "Bypassing backdoor detection algorithms in deep learning". In: *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2020, pp. 175–183.

[181] Reza Shokri et al. "Membership inference attacks against machine learning models". In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017, pp. 3–18.

[182] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[183] Anthony Sottile and other contributors. *pre-commit: A Framework for Managing and Maintaining Multi-language Pre-commit Hooks*. `https://pre-commit.com/`. Accessed: 2025-04-28. 2025.

[184] Hossein Souri et al. "Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 19165–19178.

[185] National Highway Transportation Safety Administration's (NHTSA) National Center for Statistics and Analysis. *Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey*. https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115 (visited 2022-04-20). 2015.

[186] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. "Certified defenses for data poisoning attacks". In: *Advances in neural information processing systems* 30 (2017).

[187] Emma Strubell, Ananya Ganesh, and Andrew McCallum. "Energy and policy considerations for modern deep learning research". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 09. 2020, pp. 13693–13696.

[188] Ilya Sutskever et al. "On the importance of initialization and momentum in deep learning". In: *International conference on machine learning*. PMLR. 2013, pp. 1139–1147.

[189] Mahbod Tavallaee et al. "A detailed analysis of the KDD CUP 99 data set". In: *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee. 2009, pp. 1–6.

[190] The Legislature of California. *AB-2013 Generative artificial intelligence: training data transparency.* `https://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=202320240AB2013`. 2024.

[191] The Parliament of the European Union. *High-level summary of the AI Act.* `https://artificialintelligenceact.eu/high-level-summary/`. 2024.

[192] Florian Tramèr et al. "Stealing machine learning models via prediction APIs". In: *25th USENIX Security Symposium (USENIX Security 16)*. 2016, pp. 601–618.

[193] Guido Vittorio Travaini et al. "Machine learning and criminal justice: A systematic review of advanced methodology for recidivism risk prediction". In: *International journal of environmental research and public health* 19.17 (2022), p. 10594.

[194] Andrei-Daniel Tudosi et al. "Research on Security Weakness Using Penetration Testing in a Distributed Firewall". In: *Sensors* 23.5 (2023), p. 2683.

[195] A. M. Turing. "Computing Machinery and Intelligence". In: *Mind* 59.236 (1950), pp. 433–460. ISSN: 00264423. URL: `http://www.jstor.org/stable/2251299`.

[196] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. "Clean-label backdoor attacks". In: (2018).

[197] Angelos Tzotsos and Demetre Argialas. "Support vector machine classification for object-based image analysis". In: *Object-Based Image Analysis*. Springer, 2008, pp. 663–677.

[198] T V. Lazovskaya, D A. Tarkhov, and A N. Vasilyev. "Parametric neural network modeling in engineering". In: *Recent Patents on Engineering* 11.1 (2017), pp. 10–15.

[199] Aimee Van Wynsberghe. “Sustainable AI: AI for sustainability and the sustainability of AI”. In: *AI and Ethics* 1.3 (2021), pp. 213–218.

[200] Vladimir Vapnik, Esther Levin, and Yann Le Cun. “Measuring the VC-dimension of a learning machine”. In: *Neural computation* 6.5 (1994), pp. 851–876.

[201] Lingfeng Wang and Kay Chen Tan. “Software testing for safety critical applications”. In: *IEEE Instrumentation & Measurement Magazine* 8.2 (2006), pp. 38–47.

[202] Siyue Wang et al. “Defensive dropout for hardening deep neural networks under adversarial attacks”. In: *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE. 2018, pp. 1–8.

[203] Xianmin Wang et al. “The security of machine learning in an adversarial setting: A survey”. In: *Journal of Parallel and Distributed Computing* 130 (2019), pp. 12–23.

[204] Yu Emma Wang, Gu-Yeon Wei, and David Brooks. “Benchmarking TPU, GPU, and CPU platforms for deep learning”. In: *arXiv preprint arXiv:1907.10701* (2019).

[205] Zhibo Wang et al. “Threats to training: A survey of poisoning attacks and defenses on machine learning systems”. In: *ACM Computing Surveys* 55.7 (2022), pp. 1–36.

[206] GR Weckman, RL Shell, and JH Marvel. “Modeling the reliability of repairable systems in the aviation industry”. In: *Computers & industrial engineering* 40.1-2 (2001), pp. 51–63.

[207] Jonathan Wickert and Kemper Lewis. *An introduction to mechanical engineering*. Cengage learning, 2013.

[208] Huan Xu, Constantine Caramanis, and Shie Mannor. “Robustness and Regularization of Support Vector Machines.” In: *Journal of machine learning research* 10.7 (2009).

[209] Weilin Xu, David Evans, and Yanjun Qi. “Feature squeezing: Detecting adversarial examples in deep neural networks”. In: *arXiv:1704.01155* (2017).

[210] Jenny Yang et al. “Algorithmic fairness and bias mitigation for clinical machine learning: Insights from rapid COVID-19 diagnosis by adversarial learning”. In: *medRxiv* (2022), pp. 2022–01.

[211] Lynette Yarger, Fay Cobb Payton, and Bikalpa Neupane. “Algorithmic equity in the hiring of underrepresented IT job candidates”. In: *Online information review* 44.2 (2020), pp. 383–395.

[212] Fahri Anıl Yerlikaya and Şerif Bahtiyar. “Data poisoning attacks against machine learning algorithms”. In: *Expert Systems with Applications* 208 (2022), p. 118101.

[213] E Yu, L Liu, and J Mylopoulous. "A social ontology for integrating security and software engineering". In: *Integrating security and software engineering: Advances and future visions*. IGI Global, 2007, pp. 70–106.

[214] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. "Efficient Defenses Against Adversarial Attacks". In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. AISec '17. Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 39–49. ISBN: 9781450352024.

[215] Matthew D Zeiler. "Adadelta: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701* (2012).

[216] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. "Mitigating unwanted biases with adversarial learning". In: *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. 2018, pp. 335–340.

[217] Kai Zhang and Minxia Luo. "Outlier-robust extreme learning machine for regression problems". In: *Neurocomputing* 151 (2015), pp. 1519–1527.

[218] Xiaoge Zhang, Felix TS Chan, and Sankaran Mahadevan. "Explainable machine learning in image classification models: An uncertainty quantification perspective". In: *Knowledge-Based Systems* 243 (2022), p. 108418.

[219] Yifan Zhang et al. "Deep long-tailed learning: A survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.9 (2023), pp. 10795–10816.

[220] Yaoyao Zhong and Weihong Deng. "Towards transferable adversarial attack against deep face recognition". In: *IEEE Transactions on Information Forensics and Security* 16 (2020), pp. 1452–1466.

[221] Billie J Zirger and Janet L Hartley. "The effect of acceleration techniques on product development time". In: *IEEE Transactions on Engineering Management* 43.2 (1996), pp. 143–152.

[222] Fangyu Zou et al. "A sufficient condition for convergences of adam and rmsprop". In: *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*. 2019, pp. 11127–11135.

Paper I

Safety-critical computer vision: an empirical survey of adversarial evasion attacks and defenses on computer vision systems.

Charles Meyers, Tommy Löfstedt, and Erik Elmroth.

Artificial Intelligence Review, Volume 56, 2023.

To what degree can we trust modern ML models?

Safety-critical computer vision: An empirical survey of adversarial evasion attacks and defenses on computer vision systems

Charles Meyers[1], Tommy Löfstedt[1], and Erik Elmroth[1]

[1]Department of Computing Science, Umeå University, Umeå, Sweden

***Abstract*—Considering the growing prominence of production-level AI and the threat of adversarial attacks that can poison a machine learning model against a certain label, evade classification, or reveal sensitive data about the model and training data to an attacker, adversaries pose fundamental problems to machine learning systems. Furthermore, much research has focused on the inverse relationship between robustness and accuracy, raising problems for real-time and safety-critical systems particularly since they are governed by legal constraints in which software changes must be explainable and every change must be thoroughly tested. While many defenses have been proposed, they are often computationally expensive and tend to reduce model accuracy. We have therefore conducted a large survey of attacks and defenses and present a simple and practical framework for analyzing any machine-learning system from a safety-critical perspective using adversarial noise to find the upper bound of the failure rate. Using this method, we conclude that all tested configurations of the ResNet architecture fail to meet any reasonable definition of 'safety-critical' when tested on even small-scale benchmark data. We examine state of the art defenses and attacks against computer vision systems with a focus on safety-critical applications in autonomous driving, industrial control, and healthcare. By testing a combination of attacks and defenses, their efficacy, and their run-time requirements, we provide substantial empirical evidence that modern neural networks consistently fail to meet established safety-critical standards by a wide margin.**

***Index Terms*—Adversarial Machine Learning, Computer Vision, Autonomous Vehicles, Safety-Critical**

I. Introduction

Vehicular accidents, medical mistakes, and industrial safety failures are among the leading causes of preventable death around the world [35], [56], [45]. Technologies like image classification systems have shown to be more accurate than their human counterparts under strict laboratory conditions in these domains [36], [62], [8]. However, prior research has shown that machine learning systems often fail to correctly classify images after small perturbations to the original image. While these "adversarial attacks" [18], [9], define a worst-case scenario for a given data pipeline, imperfect data is a natural result of any sufficiently complex system [65]. In this work, we focus on intentional perturbations to the input space where the goal is to *evade* a classifier, but similar perturbations are a natural consequence of modern neural network architectures and hardware setups (see Section II-A). Prior research has shown that proper data sanitation, anomaly detection, and model retraining are effective ways to combat adversarial attacks [18], [41], [53], [80]. However, even state of the art defenses decrease the accuracy when compared to the undefended (control) model, suggesting that the actual ability to generalize beyond laboratory test cases has been overestimated in the literature. This has been noted before [10], [16], [80], [53] and we confirm it below (see Section VII). Furthermore, recent research [29] has shown that most defenses have worse performance against adversaries not tested at the time of publishing, arising from the tendency to only publish the 'best' results or better methods and hardware *that become* available to subsequent researchers.

Further questions about the feasibility of truly 'safe' artificial intelligence (AI) have been raised. For instance, it has been proven that no matter where we draw our boundary conditions, there exists an attack that will confuse any (non-perfect) discriminator or shift its boundary conditions [32]. Additionally, while attacks are always possible on paper, a cost-aware analysis can reveal the feasibility of such attacks in practice. It is necessary for any safety-critical system to be robust to these attacks because, as we demonstrate, many classes of attacks are reliable even when we restrict perturbations to a single byte (see Figure 11).

While we are unable to demonstrate safety-critical computer vision models, there is some remaining optimism due to techniques like network pruning [72], [28], [47], regularization [69], [46], genetically evolved neural network architectures [74], and FIRENETS [24]. However, the efficacy of these models is reported only with test-set accuracy numbers (see Eq. 1) that do not reflect the marginal computational cost of the techniques—which is consistently significant. In practical, real-time, and/or embedded systems, this is could make the technique unusable. Therefore, for a practical analysis, we need a metric that encodes both the change in accuracy and the computational cost of that change. Furthermore, the already-existing regulatory requirements for safety-critical electro-mechanical systems (see: Section II-E and Section V) require such a high degree of precision that traditional error estimation techniques (*e.g.* traditional test-train split methodology) would be impractical to evaluate for every software change, despite that evaluation being a legal necessity (see Section II-E and Section V). Furthermore, in order to estimate the confidence

Email: cmeyers@cs.umu.se
Email: tommy@cs.umu.se
Email: elmroth@cs.umu.se

region, one must evaluate the techniques across the set of feasible hyperparameters–an often neglected practice in the literature which is frequently centered around marginal gains on benchmark data [31]. This should also include any signal pre-processing techniques (see Section IV-D), any output post-processing techniques (see Section IV-D6), and any attacks (see Section VI), as well as the traditional model hyperparameter optimization.

As such, we evaluated a large suite of proposed attacks and defenses in the contexts of accuracy, worst-case failure rate, and computation time. We show that every model defense configuration reduces the *accuracy* on benign (unperturbed) data. We show that, even when a particular defense decreases the *failure rate* against a given attack, that that behavior is inconsistent across distance measures and attack types. Most importantly, by using adversarial attacks to estimate the upper bound of the failure rate (see Section 14), we conclude that each and every tested configuration fails to meet safety-critical standards by a wide margin.

A. Contributions

- We show that even state-of-the-art defenses fail to make models that meet safety critical standards even if they tend to marginally improve the failure rate.
- We apply time and cost analysis for both the attacks and the defenses, something rarely done in the literature.
- We provide new insight into the robustness versus accuracy problem.
- We establish a standards-based framework for testing safety-critical computer vision systems in a way that meets regulatory standards without needing an infeasible number of test images.
- We survey a large suite of attacks and defenses to examine how each defense fares against each attack, measuring accuracy, worst-case failure rate, and run-time requirements in the context of safety-critical systems.

II. Background

Machine learning, artificial intelligence, and automated data collection are increasingly used in safety-critical applications like autonomous vehicles [41], [3], medical imaging [22], [71], and industrial control [38], [58]. Convolutional neural networks (CNNs) have demonstrated unparalleled accuracy in image classification tasks; however, CNNs have been shown to be very fragile to adversarial attacks [57], [34]. Research points to societal trust in fully automated banking thanks to, among other things, verifiable transactions and guarantees from the issuing institution [1]. However, when it comes to real-time, safety-critical deep learning, models are rarely reproducible or verifiable [77], [16]. Despite the drawbacks of deep-learning, modern aviation relies on an array of sensors to make largely automated decisions, relying on a framework of component testing and simulation [67]. While similar test suites for adversarial robustness have been proposed [16], [18], they rely on *de facto* accuracy goals rather than a solid theoretical and legal framework. The problems with this are plentiful. Historically, marginal gains have relied on exponentially larger models to produce increasingly marginal gains [31]. These models rely on increasingly larger datasets [31], [79], [13], which increasingly come from fewer sources [48], leading to gender-biased models [54], racism [15], and fatal design errors [6] This is a trend that goes back decades [27], [66], [15], leading to, for example, significantly higher fatality in car accidents for female-bodied people [33] or neural networks that unintentionally encode racial information from medical imaging data alone [39]. Furthermore, data collection can be expensive [68], raises serious privacy concerns [12], increases time to market [50], and impedes development speed [86]. Furthermore, research is focused on metrics that tend to be optimistic at best [55].

A. Image Classification Systems

In general, an image classification system, K, attempts to parse some image input signal, x, and output one of k class labels, $\hat{y} = K(x)$, with $\hat{y} \in \{1, \ldots, k\}$. Each image is represented as a multi-dimensional array of $n \times m$ pixels, with bit depth b and color depth c, such that they are of size $\frac{m \cdot n \cdot b \cdot c}{8}$ bytes. When the model is a neural network, the images are passed into a composition of 'layers', each layer typically performing an affine transformation followed by a non-linear element-wise transformation (called an activation function). The free parameters of such a composition of layers (called an 'architecture') are found by minimizing a loss function, $L(y, K(x))$, that penalises differences between a true label, y, and an estimate, $\hat{y}$. When the problem is a multi-class classification, the loss function could, for instance, be the categorical cross-entropy loss [10], [77], [29], [16].

a) Adversarial vs. Benign: The accuracy is measured as either *benign* or *adversarial accuracy*. The former refers to the model performance on the original dataset (denoted unperturbed/benign/ben.) and the latter refers to a dataset generated by an attacker crafted to fool the model (denoted perturbed/accelerated/adversarial/adv.). In general, electro-industrial safety systems are governed by the International Electrotechnical Commission, IEC 61508 [26], and medical software in particular requires continuous failure rate testing adding a massive computational burden to the development phase as governed by IEC 62034 [25]. In general, acceptable risk is expressed as a matrix (see Table I) where these classes are known in the standard as the safety integrity level (SIL), which then corresponds to different failure modes for components that act on-demand (*e.g.*, medical imaging) or ones that act continuously (*e.g.*, object detection in autonomous cars). In general, for safety-critical systems, we aim for SIL levels III or IV, corresponding to failures that lead to injury or death respectively. Additionally, SIL levels I or II are generally considered to be unacceptable. In the context of safety-critical systems, whether the component be hardware or software, each component must meet the requirement in isolation, raising questions of legal compliance for any system that relies on proxy models, attack detection, or any other type of out-of-band component to ensure safety.

B. Adversaries

In general, an *attacker* seeks to maximize the loss against a given model rather than to minimize it. This is accomplished by perturbing samples from one class so that they fall within the highly confident region of another, incorrect class. That is, attacks, by definition, are the worst-case perturbations of a given sample for a given model. While the literature focuses mainly on intentional adversaries, we posit that small perturbations of the input space are inevitable given the nature of real-world systems and that adversarial attacks simulate these failures. That is, things like calculation error, lens flare, lens aberration, dust, sensor failure, low-light conditions, and precipitation will all create noise that could inadvertently become adversarial. So, in an effort to measure and minimize these failures, we evaluate models against several possible attacks that attempt to *induce* different types of failures (see Section III), each of which is subject to its own optimization criteria.

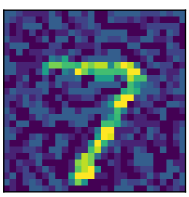


Fig. 1. A '7' perturbed with adversarial noise such that the model perceives it to be a '2.'

a) Attack Distance: Figure 2 depicts a radial basis function support vector machine, classifying the points into orange (class 0) or blue (class 1). However, the red points indicate successfully generated adversarial examples. Figure 1 shows an example of a '7' that has added adversarial noise such that the classifier sees it to be a '9'. In related surveys [32], [9], [7], [18], [16], [29], a model's adversarial robustness is defined as its performance accuracy against a given adversary. A thorough examination of such attacks are explored in Section III below, but, in general, an attacker perturbs an image such that the perturbation distance, d, is less than or equal to a threshold value, ϵ, specified by the experimenter, under some conception of distance. The evaluations in this study examine both the ℓ_∞ and ℓ_2 norms for the gradient-based attacks and ℓ_1 in other contexts, which consider perturbations no larger than ϵ, where ℓ_∞ and ℓ_2 allow for perturbations across the entire feature space, and ℓ_0 restricts the number of perturbed features. In our study, we restrict this perturbation distance to be 1 byte, as is typical in the literature [55], [57], [10], [11].

b) Attack Strength: Since adversarial perturbations, by definition, are the perturbations that maximizes the model loss through various methods, each approximates a different worst-case scenario. The 'strength' of an attack is generally related to the magnitude of these perturbations [16], and is measured by retrospective evaluations of model accuracy in which a 'strong' attack induces more loss. It is necessary to evaluate against the strongest possible attack for a given model and data set, but the strength of an attack is always contextual, since the magnitude of a perturbation must be measured with respect to some normed vector space, is specified in advance, and subject to real-world constraints. Additionally, we know that models optimized to prevent one attack do not necessarily generalize to other attacks [16], especially across distance metrics.

C. Defenses

The attacks outlined above are capable of breaking state of the art image recognition models. However a variety of defenses have been proposed that act on the dataset or the model output. Those broadly fall into categories that seek to identify an attacker and isolate them from the model API, ones that seek to isolate tainted examples from the database, or ones that attempt to mitigate all potential attacks during run-time. Below, we outline a wide variety of defenses proposed over the last several years and measure their effect on the failure rate of various models. We have excluded model transformation defenses and secondary detection models since those sidestep the problem of making a given model more robust and do nothing for the IEC requirement that each electro-mechanical component meets the regulatory standard in isolation from each other component (see Section V).

D. Attacker's Knowledge

While the general assumption is that an attacker has whitebox access to an entire pipeline (including data, model weights, and output), that does not necessarily need to be the case [20], [18]. While some attacks do need whitebox access, prior research [20], [60] has shown that a surrogate model and data-set can be used to approximate K using a proxy model $\widehat{K}$, built using the class labels provided by the model at test-time, that is sufficient for creating strong adversarial examples. Tramèr *et al.* [76] examined popular machine learning as a service platforms that return confidence values as well as class labels, showing that an attacker can build a proxy model by querying $p+1$ random $p-$dimensional inputs for unknown $p+1$ parameters. Further research [37] was able to reverse engineer the training data-set through black-box attacks against a model that returns confidence levels, with the caveat that the inferred data might be a meta-prototypical example that does not appear in the original dataset.

Fortunately for our attacker, such examples are still useful for determining the underlying data distribution, even if they manage to preserve some of the privacy of the original dataset. Shokri *et al.* [73] presented a membership inference attack that determines whether a given data point belongs to the same distribution as the original training data using a set of proxy models. Below, we examine the efficacy of several attacks from the perspective of loss as well as the functional query bandwidth.

E. Metrics

[Note: This subsection has been moved. It was previously before the adversary and defence sections.]

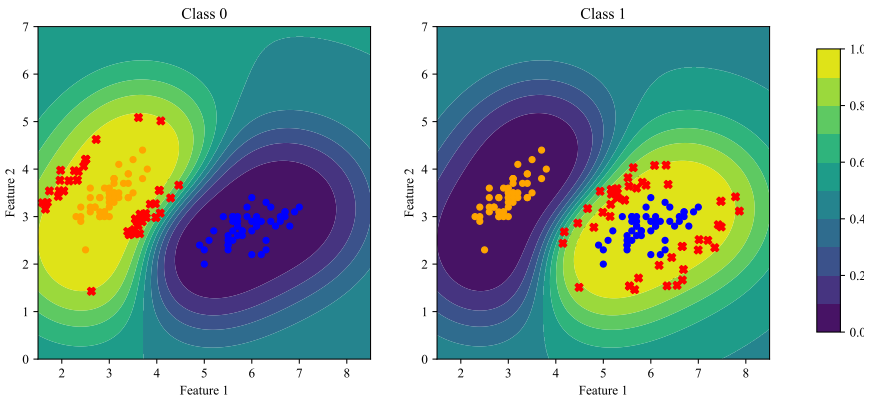


Fig. 2. Orange (class 0) points, blue (class 1) points, and red (adversarial) points in a contour map for a radial basis function support vector classifier. The contours reflect the confidence levels for a given sample and class. The bright yellow regions indicate areas of strong positive confidence and the purple areas indicate strong negative classification, and grey represents an uncertain classification. As we can see, it is rather trivial to shift the classification of a given sample towards the ambiguous regions (teal colored).

The ISO standards [61] define the Safety Inegrity Level (SIL) in failures/per hour, which we have converted to failures per second in Table I. If assume that *accidental* adversarial errors are possible in real-world systems due to things like dust, lens flare, component failure, packet loss, *etc.*, it naturally follows that the adversarial failure rate is an estimate of the models behavior at the edge or in the 'worst-case scenario'. That is, the *adversarial failure rate* is an estimate of the upper bound of the real-world failure rate in adverse but otherwise mundane circumstances.

a) Accuracy: The accuracy is defined as

$$\text{Accuracy} = 1 - \frac{\text{False Classifications}}{\text{Total}} = 1 - \eta, \quad (1)$$

where *Total* is the number of tested samples, *False Classifications* refers to the number of objects that were incorrectly categorized by a given model[, and η is the generalized error rate. In practice, η is generally assumed to be the accuracy on a 'test set', with samples from a distribution assumed to be identical to the training set. Elsewhere, we refer to this 'test set' accuracy as the 'benign' accuracy or with the subscript 'ben.' such that the test accuracy is $\eta_{ben.}$. In addition to this metric, we include metrics for a variety of signal processing techniques (see: Section IV) where the unaltered signal is designated as 'control'. Finally, we include many sets of test sets specifically crafted to be 'adversarial' (see: Section III), which are denoted with the subscript 'adv.'.

However, due to the large number of samples required by regulatory standards and the strenuous testing requirements of safety-critical software (see Section V), these evaluations become an infeasible way to verify that a model only fails once across the required number of samples (see Table I), especially if we would like to be highly confident of that estimation.

TABLE I
ACCEPTABLE FAILURE RATES FOR DIFFERENT SIL LEVELS IN WHICH A SINGLE DEATH IS POSSIBLE, MEASURED IN FAILURES PER SECOND.

SIL	On-demand Operation	Continuous Operation
I	$[10^{-6}, 10^{-5})$	$[10^{-10}, 10^{-9})$
II	$[10^{-7}, 10^{-6})$	$[10^{-11}, 10^{-10})$
III	$[10^{-8}, 10^{-7})$	$[10^{-12}, 10^{-11})$
IV	$[10^{-9}, 10^{-8})$	$[10^{-13}, 10^{-12})$

b) Failure Rate: Instead of evaluating every software change in our pipeline against the legally required $[10^7, 10^{12})$ number of samples, we can measure the precise failure rate (elsewhere λ) with a much smaller number of samples if we measure it with

$$\text{Failure Rate} = \frac{\text{False Classifications}}{\text{Total Time (s)}} = \lambda, \quad (2)$$

where *False Classifications* is the number of misclassified samples, and *Total time* refers to the total time it takes to classify all the samples.

F. Robustness

Robustness, then, is a measure of how well a model resists these *induced* failures. In this survey, we examine how several different model and data transformations (see Section IV) influence this property for a given model architecture and dataset. We measure the efficacy of a given model change, using the *Percent Change in Accuracy* ($\%\Delta ACC$):

$$\%\Delta\text{ACC} = \frac{\text{Acc.} - \text{Control Acc}}{\text{Control Acc}} \cdot 100 \quad (3)$$

where Acc refers to the *accuracy* as defined in Eq. 1 and $Control$ refers to the performance of the unchanged model on the benign ($Ben.$) dataset. This measures the marginal risk of failure for a particular model change (defense) in the adversarial case when compared to the benign case. We also defined the the metric *Relative Change in Failure Rate ($\Delta\lambda$):*

$$\Delta\lambda = \frac{\lambda_{\text{control}} - \lambda}{\lambda} \tag{4}$$

where λ refers to the failure rate, $Control$ refers to the unchanged model. Taken together, these two metrics allow us to measure the marginal risk of a given defense in both the benign and adversarial circumstances. In both cases, a positive number indicates an improvement in relative risk and a negative number indicates a worsening of relative risk, Eq. 3 in the context of accuracy and Eq. 4 in the context of failure rate.

G. Hyperparameter selection

For many attacks, hyper-parameters such as the targeted false confidence threshold, step size, batch size, number of iterations, and distortion norm must be specified in advance by the attacker [16]. Furthermore, because many of these are drawn from a continuous (and therefore infinite) space, finding a strong attack is computationally expensive and finding the strongest possible attack is at least NP-Hard [16], with the problem exacerbated by the extreme non-linearity of CNNs. Even more concerning, there is not yet a mathematical foundation for what constitutes a 'good' attack, relying only on after-the-fact evaluations of model accuracy. By examining a large hyper-parameter space, we demonstrate how each defense generalizes across the feasible attack spectrum rather than relying on a single canonical evaluation metric.

H. Attack and Defense Cost Analysis

Like in cryptography, the fundamental limit for an adversary has to do with computational cost [44]. For example, model inversion attacks become pointless if it is computationally more expensive to steal a model than it is to train one. Likewise, model defenses are only as useful insofar as they have the ability to be deployed in existing real-time systems. As such, we examine the cost of various defenses as well as the number of queries and query rate of various attacks. The best modern methods are limited to computationally expensive techniques like reject on negative impact [60], Bayesian subset analysis [7], and model post-processing techniques that degrade accuracy with added computational cost [5], [75], [53], making them unsuited for the task of improving our ability to reliably generalize. For most attacks, we tested the perfect knowledge scenario, but we have also included the 'HopSkipJump' attack [20] to model the worst case for an attacker that only has access to a standard machine learning application programming interfaces (APIs) which only exposes hard class labels.

III. Attacks

The following section outlines a variety of attacks, broken into three categories: gradient-based attacks, gradient approximating attacks, and universal attacks. For the sake of the reader, a collection of generated adversarial samples follows the mathematical descriptions in the subsection following the aforementioned trio (see: Sec. III-D).

A. Gradient-Based Attacks

The seminal work on adversarial attacks in the context of modern neural networks was written by Madry et al. [55] and an important follow up work was written by Carlini and Wagner [16]. Both operate under the condition that the attacked model has a gradient that is known to the attacker. Each of these can be considered *white-box* attacks because they are given access to the model output and the model gradient.

1) FGM: The fast gradient sign method (FGM) [40] is the most basic and fastest such attack. It is defined by the step,

$$\widetilde{x} = x + \epsilon \cdot \operatorname{sgn}\Big(\nabla_x L\big(y, K(x)\big)\Big),$$

where L is the loss function, as described above, $\nabla_x L$ is the gradient of L with respect to the input x, the ϵ is the maximum perturbation distance, sgn is the element-wise signum function, and $\widetilde{x}$ denotes a generated adversarial sample. It is called 'fast' because it does not check the feasibility of the perturbation (if it is within a maximum distance, ϵ, as is done in other methods, see *e.g.* PGD below).

As such, it may not be as successful as other methods, but operates very quickly. We tested several different step sizes and norms for this method, enumerated in Figures 11 and 18.

2) PGD: Projected gradient descent (PGD) [16] takes a gradient step to increase the loss, but also includes a projection step, Proj_ϵ, that enforces the constraint of a perturbation distance q of at most ϵ with respect to some norm by projecting onto the feasible set (defined by the perturbation distance, ϵ). The iteration scheme of PGD is

$$x^{(s+1)} = \text{Proj}_\epsilon\Big(x^{(s)} + r \cdot \nabla_{x^{(s)}} L\big(y, K(x^{(s)})\big)\Big),$$

where r is a step size, s is a sequence index, and $x^{(S)}$ denotes a generated adversarial sample after S steps. This iteration is repeated until a specified number of iterations, S, have been reached. The number of iterations is specified by the attacker, which ultimately determines the processing-time against a given model, scaling linearly with the number of iterations. We tested several different step sizes and norms for this method, enumerated in Figures 11 & 18.

3) Carlini–Wagner: Carlini and Wagner (CW) [16] devised an attack that minimizes the perturbation distance subject to a distance constraint while maximizing the false confidence. The iteration scheme is for some constant, C,

$$x^{(s+1)} = \arg\min_x \|x^{(s)} - x\|_2^2 + C \max\big(\max_{j \neq t} g_j(x) - g_t(x) + C, 0\big)$$

where the g_j, g_t are discriminant functions and the goal is to minimize the perturbance with a penalty for not changing it to the target class. It also generalizes beyond the squared ℓ_2 norm. This method attempts to enforce attack quality by

penalizing examples with low confidence and continuing to iterate on them until either a maximum number of iterations or the specified false confidence is reached. For our tests, we used the ℓ_∞ norm and a confidence threshold of 99%.

B. Gradient-Approximating Attacks

Further work sought to find attacks that did not rely on explicit gradient information. The *Deepfool* attack [59] uses a quadratic approximation of the boundary condition to generate a separating hyperplane, while *Few-Pixel* and *Threshold* [49] use a search algorithm known as differential evolution rather than relying on explicit gradient information. Each of these attacks can be considered *grey-box* attacks because they rely on un-categorized model output but use that output to approximate the gradient, rather than rely on explicit access to the model weights.

1) DeepFool: The DeepFool attack [59] seeks to find the smallest perturbation that causes a misclassification using a first-order Taylor approximation of the classifier. In essence, it seeks to find the minimal separating hyperplane between the target sample's true class and another. It assumes that it has access to the entirety of the model, including the confidence level (expressed as a logit) of all labels, the true label of a sample (x) and the model gradient. The iteration scheme is

$$x^{(s+1)} = \arg\min_x \|x^{(s)} - x\|_2$$

subject to

$$f(x^{(s)}) + \nabla_{x^{(s)}} f(x^{(s)})^T (x^{(s)} - x) = 0,$$

with $\|\cdot\|_2$ is the ℓ_2 norm, and where f outputs the logits of K, such that $K(x) = \phi(f(x))$, with output activation, ϕ. Runtime is determined largely by the number of iterations, S, and the number of gradients $w.r.t.$ $L(x)$ used to estimate $f(x)$, which we set to 10 such that the gradient was calculated for each class. The existence of any such attacks with a distance than or equal to some some specified robustness threshold ($\epsilon \leq \epsilon_{critical}$) should immediately cause concern.

2) Few-Pixel: The few-pixel attack [49] (Pixel) attempts to maximize the loss by iteratively finding the least robust pixel set and perturbing it by less than some specified threshold, ϵ. That is, this attack seeks to maximize false confidence while minimizing the number of perturbed pixels. The iteration scheme is

$$x^{(s+1)} = \arg\max_r L\big(y, K(x^{(s)} + r)\big) \quad \text{subject to} \quad \|r\|_0 \leq \epsilon,$$

where r is the perturbation and ϵ is the perturbation distance in pixels, typically of just one pixel. This attack uses differential evolution to simultaneously optimize for both loss and the number of perturbations. In our case, we limited this to a single perturbation, but did not restrict epsilon beyond the normal $[0, 255]$ range. Since perturbation distance for a single pixel is not restricted, we restricted the number of distorted pixels to one of these: $[1, 2, 4, 8, 16]$. Unlike the aforementioned gradient-based attacks, this method generalizes to classification systems that lack gradients (*e.g.*, decision trees). This differs from the the *Threshold* and *Adversarial Patch* attack below by optimizing for the fewest number of changed pixels (ℓ_0 norm) rather than the perturbation distance (ℓ_2 or ℓ_∞).

3) Threshold: The threshold attack [49] is similar to the few-pixel attack (in that it uses differential evolution as the optimization algorithm), but uses the ℓ_∞ norm rather than the ℓ_0 distance. This method, like the Carlini Wagner (CW) method, attempts to generate examples that are both false and highly confident. However, it uses a complicated algorithm (differential evolution) rather than simple, linearized methods (as in CW). The iteration scheme is

$$x^{(s+1)} = \arg\max_r L\big(y, K(x^{(s)} + r)\big) \quad \text{subject to} \quad \|r\|_\infty \leq \epsilon$$

where ϵ is the targeted perturbation threshold of .03. Unlike the aforementioned gradient-based attacks, this method generalizes to classification systems that lack gradients (*e.g.*, decision trees). This differs from the the *Few-Pixel* and *Adversarial Patch* attack below by optimizing for largest change in loss (ℓ_1 norm) rather than fewest number of pixels (ℓ_0 norm) or the perturbation distance (ℓ_2 or ℓ_∞).

C. Universal Attacks

Even further work has sought to go beyond computationally-intensive approximations for each attacked sample or model query. The *Adversarial Patch technique* [14] uses the aforementioned differential evolution algorithm to generate an additive noise sample that maximizes the loss **for all samples** and is considered a *grey-box* attack because it relies on un-categorized model outputs but not explicit access to the model weights. However, we include it here because the nominal image patches generated by this technique are meant to generalize to unseen data as the number of API queries increases, meaning that this can be trained using data wholly disconnected from the model-builder's dataset. The *HopSkipJump* attack [20] has been shown to minimize the number of API queries required to break **any model**. As such, it is considered a *black-box* attack.

1) Adversarial Patch: The Adversarial Patch attack [14] (Patch) uses the same differential evolution algorithm as the Threshold Attack. However, instead of optimizing for a threshold or confidence level, it seeks to change each class into any other by applying an image patch that is not unique to a given image, but is universal for a given dataset. This single generated image patch is added to each image and is intended to cause a generic misclassification, regardless of the original class. We know that these attacks are quite general for a given dataset and not specific to a given model [14], [82], raising serious concerns about an attacker's ability to generate universal and offline attacks for a given set of data. Like with the Pixel attack, the per-pixel distortion is not restricted, so we restricted perturbation to a percentage of the benign image, using the hyper-parameters [.03, .1, .25, .5, 1.0]. Unlike the aforementioned gradient-based attacks, this method generalizes to classification systems that lack gradients (*e.g.*, decision trees). This differs from the the *Threshold* and *Adversarial Patch* attack below by optimizing a single perturbation that works for every image rather than minimizing the geometric perturbation distance (ℓ_2 or ℓ_∞ norm) while

minimizing the number of pixels required for said universal patch (ℓ_0).

2) HopSkipJump: The HopSkipJump attack [20] (HSJ) is a query-cost-aware model that acts on hard class labels rather than confidence levels (*i.e.*, a blackbox attack). It finds a point on the boundary (using binary search between the initial attacked point and a point on the other side of the decision boundary), and approximates the gradient at the boundary using Monte Carlo sampling in an offline manner. Then, a step is taken in the approximated gradient direction, the model is queried again with the new points, and the process is repeated. The procedure iterates the step,

$$x^{(s+1)} = \mathrm{Proj}_{\epsilon, x^{(0)}}\left(x^{(s)} + r^{(s)} \frac{\widetilde{\nabla}_{x^{(s)}} Q(x^{(s)})}{\left\|\widetilde{\nabla}_{x^{(s)}} Q(x^{(s)})\right\|_2}\right),$$

where $x^{(s)}$ is an adversarial sample, s a sequence index, $\mathrm{Proj}_{\epsilon, x^{(0)}}$ is an ℓ_2 projection of a point onto a sphere of radius ϵ, centered at the initial point $x^{(0)}$, and $\widetilde{\nabla}_x Q(x)$ is the Monte Carlo estimate of the gradient of Q at x, and

$$Q(x) = \max_{c \neq c^*} K_c(x) - K_{c^*}(x),$$

where K_c is the model output for class c, and c^* is the class of the initial point, $x^{(0)}$. This attack's run-time is controlled by the number of (offline) gradient estimations, the number of random samples per (online) query, and the number of iterations of both parts of the algorithm. This attack finds an adversarial example that is close ($\epsilon \leq \epsilon_{critical}$) to the original sample, but still causes a misclassification. This model is query efficient as it uses a small number of API queries to approximate the gradient and then uses this information to approximate the gradient near the class boundary and optimize the perturbation distance of the attacked sample. Also, because this attack approximates the gradient rather than relying on explicit whitebox access to it, it can be used to attack non-differentiable models (*e.g.* random forests). That is, this is one of the most universal black box attacks.

D. Attack Samples

To aid the reader, we have visualized a single attack sample for each attack on the undefended (control) model, depicted in Figures 3- 10. Additionally, we have provided the failure rate, η, defined in Eq. 2, and tested on 100 samples from the MNIST dataset. For each attack, we also vary a distance parameter, determined by the optimization criteria for each attack outlined above. For the sake of clarity different distance metrics are denoted with a distance of $epsilon$ subject to some norm, $|\cdot|_n$ such that ℓ_2 norm of ϵ is $|\epsilon|_2$

IV. Defenses

A. Attacker Identification

We must assume that at least some of the user inputs will 'adversarial', even if that adversary is sensor failure and not an intentional attack. Identifying and isolating this adverse input may not require a perfect anomaly detection system, but could draw from graph theoretical representations to identify and isolate networks of distributed attackers, allowing the model API provider to revoke access or otherwise isolate the attack effects from the models. While this has been done in the context of social networks [30], these techniques can easily be fooled with intermittent attacks [63], distributed attacks [4], or something as simple as a quadratic approximation of the model [20]. For web services more generally, legitimate users are often identified by using CAPTCHA [2], but that it not a solution for an API meant to be accessed by software. Furthermore, outsourcing this to a secondary component would run afoul of the IEC requirement that each component meet regulatory standards in isolation from all other components (see Section V). However, even if we assume all samples are generated by legitimate users with guaranteed data integrity, we still cannot be confident that 'adversarial' noise will not be generated inadvertently by routine phenomena like sensor failure, dust, low-light conditions, lens aberration, precipitation, or another mundane cause. One possible approach is to eliminate 'bad' samples at run-time.

B. Subset Analysis

Subset analysis [64] examines how a particular sample changes the model's performance [64]. By exhaustively comparing the model accuracy on various subsets of data, it attempts to isolate adversarial samples by removing ones that lead to worse-performing models (*i.e.* the sample is 'bad'). If the database is large, this becomes an incredibly expensive task. This method also assumes that all 'bad' data is, in fact, adversarial and not a legitimate measurement of real world circumstances. Another method, called 'Subset Scanning' [23], examines the hidden layers of a neural network to ensure that a particular sample looks 'typical' as it passes through the models layers rather than just at the final layer. This comes with the added cost of tracking the model through each of these layers for each of these samples, which becomes infeasible in real-time systems due to the size and complexity of neural networks and their associated datasets.

C. Attack Mitigation

Rather than relying on a generic framework for detecting and preventing all attacks, as discussed above, there are mathematical foundations for avoiding the impacts of adversarial attacks during model creation. These are either 'pre-processing' defenses or 'post-processing' ones in which alter either the data (pre-processing) or the model output (post-processing) to mitigate the risk of an attack. In general, the goal of these defenses is to reduce the noise in the input, or to reduce the precision in the output, corresponding to the pre- and post- techniques. In this way, we seek to examine how modern neural architectures perform on the generalized 1-byte spherical perturbation that surrounds a true example of a given class, rather than rely on external components to identify and mitigate the attacker.

D. Pre-processing

There are a variety of ways to change the data before training so that the resultant model is more robust to adversarial perturbations.

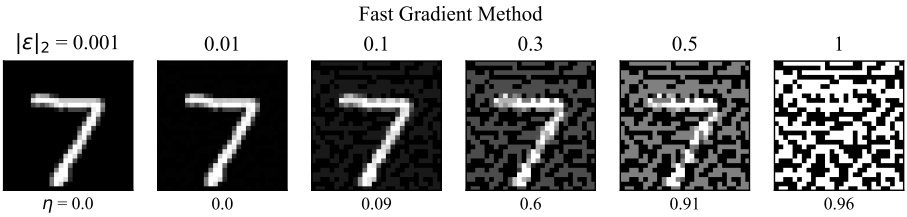


Fig. 3. The Fast Gradient Method (FGM) doesn't enforce any kind of feasibility criteria, resulting in salt-and-pepper noise as the ℓ_2 approaches the standard deviation of the normally distributed dataset.

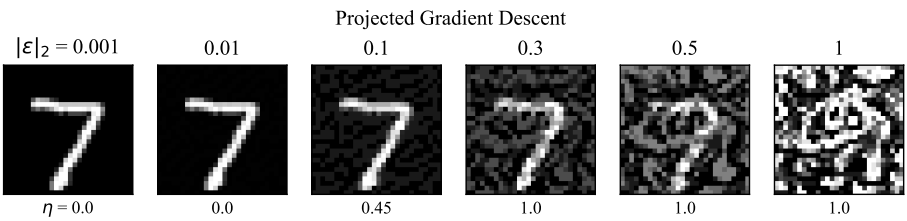


Fig. 4. The Projected Gradient Descent Attack (PGD) projects the adversarial example back onto a sphere of a fixed radius, yielding noise that more closely approximates handwritten digits.

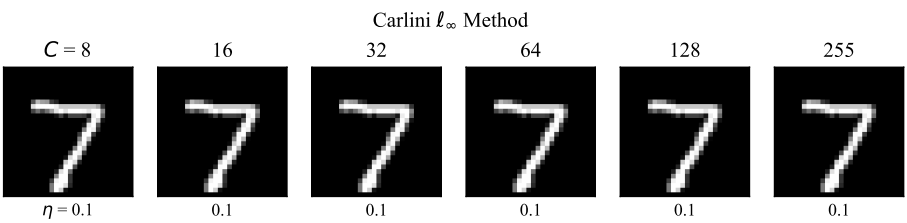


Fig. 5. The Carlini-Wagner ℓ_∞ method includes an added confidence constraint that only returns an adversarial example $\iff L(x^{s+1}) - L(x) \geq C$. Depicted here is the unperturbed 7 because this method was unable to find an adversarial example for these constraints. Despite this, we can see that for 10% of the samples are consistently misclassified with a high degree of false confidence.

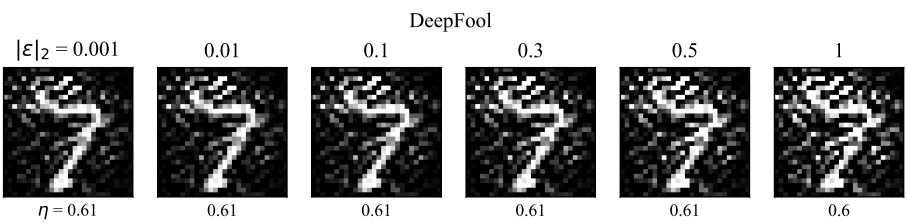


Fig. 6. The DeepFool Method, rather than constraining perturbations to an ϵ-sphere around the sample, finds perturbations that extend beyond the approximated boundary by a distance of at least ϵ. Regardless of the distance constraint, DeepFool is very effective.

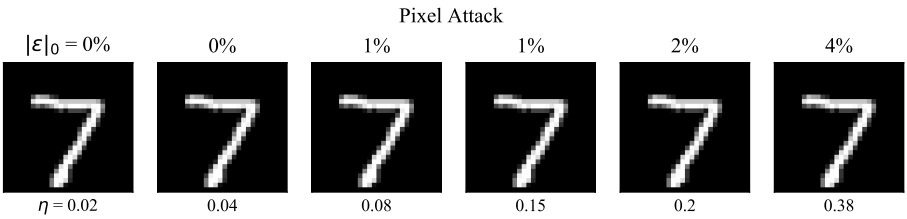


Fig. 7. The Pixel Attack (Pixel) seeks to maximize the loss while minimizing the number of perturbed pixels. Here we see that even a small perturbation in original image can lead nevertheless consistently induce failures. Here, ϵ_0 denotes the ratio of perturbed pixels to the total.

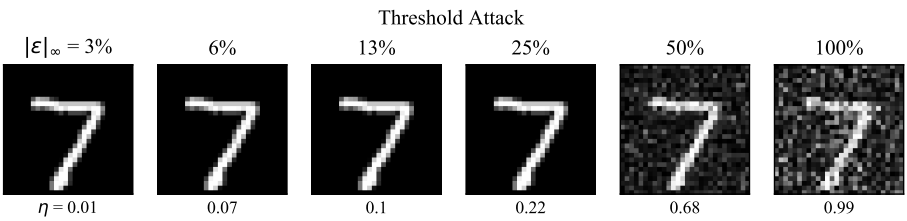


Fig. 8. Rather than minimizing the number of pixels, the Threshold Attack (Thresh) optimizes for the smallest perturbation possible. Here, ϵ_∞ denotes the ratio of the applied to noise to the maximum possible value (255).

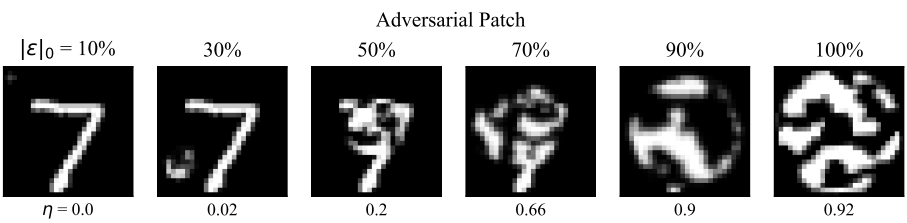


Fig. 9. This is one of the two most dangerous attacks as it can consistently find image patches (the noisy circle depicted above) from only a small number of samples ($n = 100$) that can often fool the classifier, regardless of original image or class. Obviously, as we replace all of the pixels with adversarial noise, the classifier becomes mostly useless, but this attack is still concerningly effective when only a fraction of the image has the adversarial noise. Here, ϵ_0 denotes the ratio of perturbed pixels to the total.

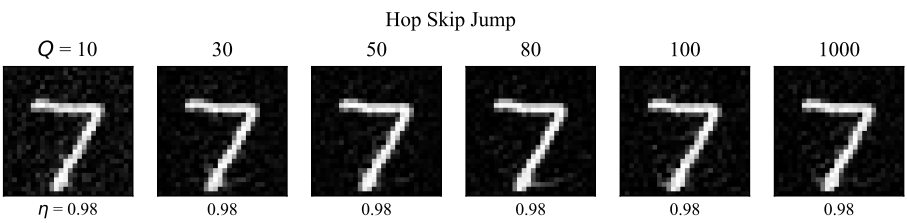


Fig. 10. The Hop Skip Jump Attack (HSJ) uses a second-order approximation of the classification boundary in an offline manner to find adversarial examples that first maximize loss then minimize for perturbation distance. This attack is uniquely concerning because it 1) works on any model and 2) is surprisingly effective even when our attacker is restricted to a small number of queries per batch (here, denoted as Q).

1) Gaussian Augmentation: The most straight-forward defense (Gauss-In) is where random Gaussian noise is added to the input data and the model trained without modifying the class labels [84]. If we replace real samples with noisy ones, the processing time and space are marginal. We tested noise with standard deviations of .9, .99, and .999 on data that was zero-centered and normalized.

2) Label Smoothing: The label smoothing (denoted Label) defense [81] sets a cap on the confidence level for a given model output. If the output layer outputs a number higher than this cap, it uniformly distributes the difference across all classes. In this way, it obscures the model output, reducing the effective query rate for the attacker. In our experiments, we set this threshold to be 99%, 99.9%, or 99.99% which itself is far below the regulatory standards outlined in Section V.

3) Feature Squeezing: The feature squeezing (FSQ) method [83] reduces the bit depth of the input image to a specified value, treated as tunable parameter, which hopefully increases the signal to noise ratio. The initial processing time merely requires setting some bits to zero which can be vectorized and parallelized, and scales with image size. However, the resulting model can use smaller data-types and potentially operate faster and require less memory. We tested bit-depths of 32 and 16 (the control is 64 bit and the images are 8 bit).

4) Total Variance Minimization: Total variation minimization (TVM) is an image de-noising techniques that dates back decades [70]. It exploits the fact that images with spurious details have high total variation. This defence is effective at preserving edges within an image, and encourages spatial homogeneity such that large jumps in intensity between neighboring pixels are penalized, leading to a smoother image, determined by some specified noise level. This minimization problem is non-trivial, and there are several specific and tailored algorithms for it [19], [42]. Thus, we tested several combinations of the noise level (denoted 'prob), enumerated in Figures 11 and 18.

5) Adversarial Re-training: Adversarial retraining (Retrain) is a method proposed by Croce *et al.* [77], that appends adversarial examples to the training set, labels them 'adversarial' and trains a classifier on the new (larger) dataset. The first problem with this method is that the training time increases linearly with the number of re-training epochs, with twenty retraining cycles being recommended in the original paper. Furthermore, 'adversarial re-training' must be conducted against each type of attack individually since the topological characteristics of attacks vary widely. An extension seeks to encode ambiguity between an adversarial example and both the original and target class, called 'confidence-calibration' [29] by changing the class label from an integer to a float that decays with distance from the 'true' image. While it offers improved results over other types of adversarial retraining, it optimizes against a particular type of failure which inherently degrades performance against others [16].

6) Post-processing: There are a variety of ways to change the the model outputs so that the user-exposed API reveals less information to the attacker.

a) Gaussian Noise: This post-processing defense (Gauss-Out), like its pre-processing counterpart adds Gaussian noise, but in this case, it applies it to the model outputs [51]. Since it acts on a discrete output vector, the marginal cost is negligible in comparison to image processing. However, it's efficacy is tied to reducing the accuracy of the API by an amount proportional to the variance of the added noise. That is, it reduces the number of useful output bits available to an attacker (as well as legitimate users).

b) High Confidence Thresholding: Rather than decrease the precision of the output as in other techniques, this method (Confident) only returns model output if the confidence level exceeds some threshold specified by the model builder [21]. While this does make it harder for an attacker to calculate a gradient step, the attacker can circumvent it by taking a step large enough to overcome the thresholding (*i.e.*, by changing the model output by more than twice the cut-off value, ensuring that the classification 'jumps over' the obscured boundary) HopSkipJump is a query efficient model for doing exactly this [20], but any other method could be effective by simply increasing the perturbation threshold such that perturbation extends beyond the 'fuzzy' class boundary. In our experiments, we set this to be equal to 50%, casting the normal one vs. one problem into one vs. the sum of the rest, ensuring that the transformed model returns no classification if the confidence of the label does not exceed the total confidence of the other labels. To simulate scenarios in which low-confidence classifications are merely ignored (which, in practice, could mitigate an attacker as they generate their attack), we exclude all such samples from the accuracy calculations for this defence.

c) Rounded: Instead of obscuring the output data with noise or only reporting highly confident answers, this method (Rounded) merely reduces the bit depth of all the reported confidence levels [85]. Instead of a 64-bit output vector, this might be reduced to an 8-bit vector reducing the effective attack query rate by a factor of four. However, this harms the legitimate user by the same degree by reducing the precision of their queries as well. For our study, we used several different numbers of decimals, reflecting the number of base 10 digits in the set $[0, 1]$ revealed by the API. enumerated in Figures 11 and 18.

d) Reverse Sigmoid: The Reverse Sigmoid (denoted Sigmoid) defense [52] changes the activation function from the rectified linear unit (ReLU) or the Weiserstrass σ-function to a function that retains the approximately linear behavior when the confidence is near 0 but instead of asymptotic convergence, model confidence eventually decreases rather than converging to one. The Reverse Sigmoid activation function, A, is defined as,

$$A(y_i) = \alpha^i(y_i - \beta(\sigma(\gamma\sigma^{-1}(y_i) - 1/2))),$$

where α, β, γ are scaling parameters, specified as hyperparameters, y_i is the class label of sample i, and σ is the logistic sigmoid function. Run-time requirements are basically identical to other activation functions as the goal of the model builder is to remain more-or-less along the linear section of the function. This has the effect of preserving $y = f(x)$ while

ensuring that the class label, $\hat{y}, \neq K(x)$. In essence, this traps an attacker in a local minimum that is in the opposite gradient direction of the global minimum. Even if this defense is known to an attacker, the non-bijective nature of this function makes the model non-invertible (beyond specified thresholds) to gradient descent methods. Alpha is a scaling parameter that is determined by β and γ, both of which must be positive. Both β and γ were evaluated in a grid search of the set $[.01, 1, 100]$ for each variable, yielding 9 combinations.

V. Safety Critical Computer Vision: A Framework for Robustness Guarantees

A. Safety Critical Computer Vision

Since the bulk of the literature focuses on image classification systems [32], [9], [7], [18], [16], [29], we chose to stress test them in the contexts of autonomous vehicles, medical imaging, and industrial control which are each governed by pre-existing standards across the different domains. Safety-critical software is already widely deployed in other electro-mechanical systems like vehicular braking systems [17], aviation [67], and medical implants (*e.g.*, pacemakers) [78]. The International Standards Organization provides a safety-threshold of 10^{-9} failures per second for any life-threatening situation [61] and 10^{-6} [61] for any risk of harm, required of any automotive or aviation system governed by ISO 26262 [61]. For an autonomous vehicle trying to classify objects on the road, a false negative classification of, for example, a cyclist could lead to death; whereas, a false positive detection of a cyclist would be more likely to cause braking-related injuries that are less severe. For medical imaging, a false negative classification could mean loss of life; whereas, a false positive is less likely to cause grievous harm (but likely to be lead to expensive and unnecessary additional testing). Understandably, these correspond to the differing legal and technical requirements outlined in these international standards.

B. A Framework for Robustness Guarantees

Since adversarial failure rate provides a worst-case estimate of the failure rate for a given context (see Section II-E), we propose a safety critical testing framework that (1) evaluates the benign and adversarial failure rate across several attack metrics (as dicated by the safety-requirements of the system) (2) repeats those evaluations across a feasible hyper-parameter space to estimate a confidence interval for the values in (1), and then (3) rejects any model that does not consistently meet the standards outlined in Table I as unsuitable for safety-critical systems. The limitations of this approach are discussed below in Section VIII.

a) Advantages: Because of the relatively small run-time requirements of this approach (when compared to testing against massive in-distribution test sets), this method could, for example, act as a unit test in machine learning applications rather than relying on full-system integration tests to evaluate changes to a single model, signal processing technique, data storage format, or API access mechanism. It could also be used to highlight error-prone classes or other subsets of data to reduce error or create synthetic samples. Furthermore, by isolating changes and testing them as quickly as possible, it's much easier to parse cause and effect when compared to full-system integration tests that could include many changes from many different development teams and require live and potentially dangerous systems (like cars or MRI machines) to effectively test. To further increase development velocity, we propose metrics Eq. 3 and Eq. 4 as standards for evaluating not only the efficacy of a given change, but as tools to quantify the marginal risk associated with each change, as dictated by the ISO 26262 standard [61].

VI. Experimental Methods

In this study, we evaluated the accuracy (see: Sec. 1) and the failure rate (see: Eq. 2) for a variety of attacks (see: Sec. III) and defenses (see: Sec. IV) using the methods discussed in detail in the previous section. In our experiments, we go beyond the in-distribution train/test split typical in machine learning research (Eq.1), which only highlights how a model will perform on the data we have already collected, rather than providing guarantees about future performance for the infinite and continuous space that is the real-world. For the latter, we measured the worst-case failure rate for a variety of different model defenses (see Section IV), using several attacks that define the 'worst-case' according to different contexts (see Section III). We trained one model for each dataset using the ResNet [43] architecture provided by Madry et al. as part of the "MNIST"[1] and "CIFAR-10"[2] challenges for each defense across several different hyperparameter combinations.

To generate confidence intervals, we varied both defense and attack parameters in an iterative grid search. Results across all tests are reported in Figures 4 and 8. Initial model weights and model architecture were taken from the survey by Madry et al [55]. Like in that paper (and commonly throughout the literature), we used the MNIST and CIFAR-10 datasets so that our survey can be directly compared to the wider literature. For each experiment, we trained the model for 20 epochs on the entire training set, as defined by the Tensorflow version of the datasets. Attacks were given a small computational budget of 10 iterations (100 samples, 10 iterations). For pre-processing defenses (see: Section), this included data transformations that were distinct from the original training process and for post-processing defenses, the model output varied (see: Section V-B2) relative to the survey by Madry et al. [55]. The MNIST dataset was classified using a simple toy model and the CIFAR-10 dataset was classified using a modified version of ResNet, both taken from the survey of Madry et al. [55]. Model prediction and attack times were measured using Python's 'process-time' due to the timing jitter associated with shared systems and operating system variability. The timing resolution was in milliseconds, far below the scale of training times and total attack times, making any noise negligible. After training, models were evaluated against the ten thousand unperturbed (benign) images available in the dataset according to both Equation 1 (accuracy) and Equation 2 (failure rate)

[1]https://github.com/MadryLab/mnist_challenge
[2]https://github.com/MadryLab/cifar10_challenge

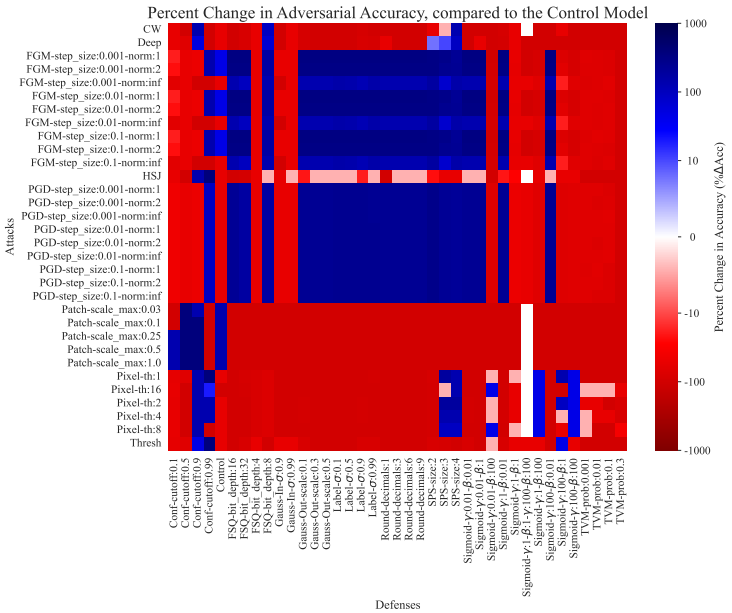


Fig. 11. The percent change in adversarial accuracy of each attack against each defense for CIFAR-10. As we can plainly see, no defense was able to improve the failure rate across all tested attacks. Red indicates that defense made a model worse. Blue indicates an improvement. White indicates no change.

in both the benign context. Each model was attacked using the same subset of 100 (disjoint) samples, randomly drawn from this set in an attempt to isolate model performance from coincidences associated with sampling. Model accuracy on this set of perturbed data is denoted as the 'adversarial accuracy' below, with 'benign accuracy' referring to the model accuracy on the unperturbed data. All experiments were run on an Intel Xeon 4210 with 32GB of memory and with an Nvidia V100 GPU in a shared-kernel environment. We evaluated hundreds of combinations of attacks and defenses as illustrated and enumerated in Figure 11.

VII. Results and Discussions

a) Benign vs Adversarial Accuracy: Figure 12 (left subplot) depicts the 95% confidence region of adversarial and benign accuracy, computed using Equation 1. The blue bars represent the accuracy on unperturbed (benign) data and the red bars represent the accuracy on perturbed (adversarial) data. From this rather large region (see Figure 12), we see that both attack and defense hyper-parameter tuning have significant effects on the accuracy of a given method, since a small change in hyperparameters can drastically change the efficacy of a given attack or defense. Figure 11 shows the percent change in accuracy between the adversarial and benign circumstances for each defense and attack. We can plainly see that some techniques fool every model nearly every time while some attacks are weak and not likely to succeed under any conditions. Since no tested configuration reliably exceeded the *benign accuracy*, this metric seems to only indicate a lower bound of the generalization error while the *adversarial failure rate* estimates the upper bound (see Section II-E). That is, we can confidently claim that our *true generalization error* (η) falls somewhere between 10^{-4} failures per second (roughly indicated by the 99.96 % test-set, benign accuracy for MNIST or 99.83% accuracy for CIFAR10) and the worst-case adversarial failure rate (roughly 10^{-1} for MNIST and 10^{2} for CIFAR-10), which obviously falls below the safety-critical standards (see Table I) by huge margins, indicating that neither architecture is safety-critical (see: Section V).

b) Defenses: In Figure 12 we see that, for every defense (left subplot), that adversarial accuracy is lower than benign accuracy, adding more empirical evidence the accuracy *vs.* robustness trade-off discussed widely in the literature (*e.g.* there robustness and accuracy are at odds with each other). The tested defenses each attempt to strategically destroy, smooth, or average data in the original dataset or the output of the model, which results in a loss of precision that makes the benign accuracy worse than the Control (see: Figure 12). Furthermore, when we examine the attacks in the right subplot of Figure 12, we see that Deep, HSJ, CW, Pixel, Patch, and Thresh are all able to confuse the model more that half of the time. Furthermore, variations in the defense performance (right side of Figures 20 and 14) raise questions about the ability of these architectures to generalize since things like bit-depth (FSQ), training-noise (Gaussss-In), label-noise (Gauss-out), and image resolution (SPS) greatly vary the failure rate. In the real-

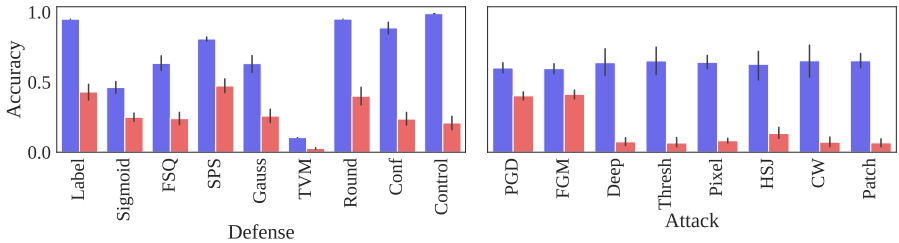


Fig. 12. The 95% confidence interval of adversarial (red) and benign (blue) accuracies for each defense (left) and each attack (right) for CIFAR-10. One trial was conducted for each hyper-parameter combination, and the confidence interval spans these trials.

world, effective resolution will change between individuals (e.g. a medical scan) or while moving (e.g. an autonomous vehicle). Even random noise drawn from approximately the same distribution as the training set (Gauss) increases the benign failure rate by an order of magnitude or two (compare the gap between *Gauss* and *Control* in both Fig. 20 and Fig. 14).

c) Attacks: Figure 12 (right subplot) depicts the same confidence region as above, but broken down by attack rather than defense. It is obvious that the Deep, HSJ, and Patch reduce the accuracy the most often. PGD, FGM, and CW are less effective, but still able to successfully perturb a significant portion of the samples. However, the Threshold and Pixel attacks are less consistent.Figure 11 demonstrates how each attack fares against each defense by depicting the percent change in accuracy. The color gradient is centered at the benign failure rate, becoming a more intense red as the accuracy decreases, with a dark red indicating substantially worse performance than with the undefended model and unperturbed data. Since no column is blue in Figure 11, no single defense is able to consistently subvert a generalized attacker, while modest gains against a particular attack are possible. Techniques like Adversarial Patch, DeepFool, and HopSkipJump consistently break models (see: Figure 11). While some defenses do provide limited protection against more advanced techniques (indicated by the color blue in Figure 11), their performance on unperturbed data tends to be reduced relative to the control (see: Figure 12, left subplot) substantial empirical evidence that the normally discussed accuracy, from Equation 1, is consistently more optimistic than what the adversarial analysis implies.

d) Computational Cost: In order to estimate computational cost, we measured each time as a process time, reducing the jitter due to operating system operations and shared kernel constraints. We see that defenses require between 1 and 100 seconds of training per success, broken down by defense in the left subplot of Figure 13. However, attacks (see right subplot of Figure 13) require as few as several milliseconds per sample in the worst case and 10 seconds in the best case, with the average attack time falling after around a half second per sample. Furthermore, we see that the Figure 14, we see how various defenses manage the trade-off between computational complexity and efficacy by measuring the failure rate as in Equation 2. The Confident model had the best accuracy (see: Figure 12) since it merely ignores queries below a certain confidence threshold, but even if we treat that as a null event (instead of a false classification), this increases the failure rate (see: Figure 14) relative to the control since objects were not detected.

e) Failure Rate: When we combine the information from the accuracy and time graphs in Equation 2, we obtain Figure 15, displaying the failure rate across attacks and defenses as well as their individualized performance. When we examine the average performance of a given defense (see: Figure 15), we see that most defenses fail to meet safety critical standards (see: Table I), even if the surveyed defenses tend to improve adversarial accuracy (see: Figure 16). Total Variance Minimization produced particularly inaccurate models which led to particularly inaccurate attacks. While some defenses do provide relief against some attacks, that behavior is inconsistent across different attacks, particularly in the case of the Patch and Pixel attacks which seem to be universally strong, even with mild perturbation constraints. Furthermore, those gains are marginal compared to the efficacy of the average attack (see: Figure 16) and the order of magnitude required by regulations (see: Figure I).

f) Attack Budget and Attacker Knowledge: When we examine the general performance characteristics of attacks, we see that the average attack takes a few seconds to induce a failure (see Figures 20 and 14). This appears to be consistent across attacks and is remarkably effective with only a small computational budget of 100 iterations and a query budget of 1000 with a perturbation distance no greater than 1 byte. This is true for both whitebox attacks (everything but HSJ) and blackbox (HSJ) attacks. Most attacks take a few seconds per sample; however, the one exception to this is the Adversarial Patch attack which takes only a few milliseconds to induce a failure (on average). Due to the universality of this attack against an entire dataset (see Section III) this failure rate would go to infinity as the sample size goes to infinity. In practice, a quadratic approximation of the boundary (HSJ) is roughly as effective as whitebox models, especially as the number of features scale (compare Figure 18 and Figure 11). However, there is a clear advantage when the attacker optimizes across a large sample of the test set (see *Patch* in Section III), with the failure rate tending towards 0 as this sample size

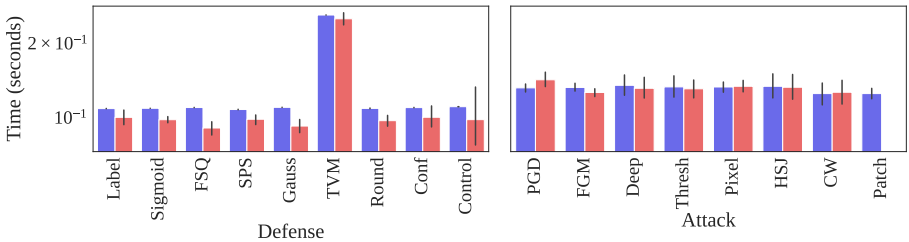


Fig. 13. The 95% confidence interval of prediction times (blue) and attack times (red) for CIFAR-10, broken down by defense (left) and attack (right). One trial is depicted for each hyper-parameter combination. Accuracy on the benign (unperturbed) dataset is depicted in blue and accuracy on the adversarial (perturbed) dataset is depicted in red.

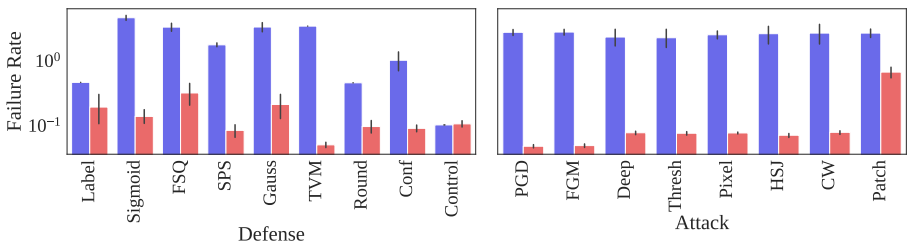


Fig. 14. The 95% confidence interval of failure rates both benign (blue) and adversarial (red) for CIFAR-10, broken down by defense (left) and attack (right). This was computed using Equation 2. The failure rate of the benign (unperturbed) dataset is depicted in blue and the failure rate of the adversarial (perturbed) dataset is depicted in red.

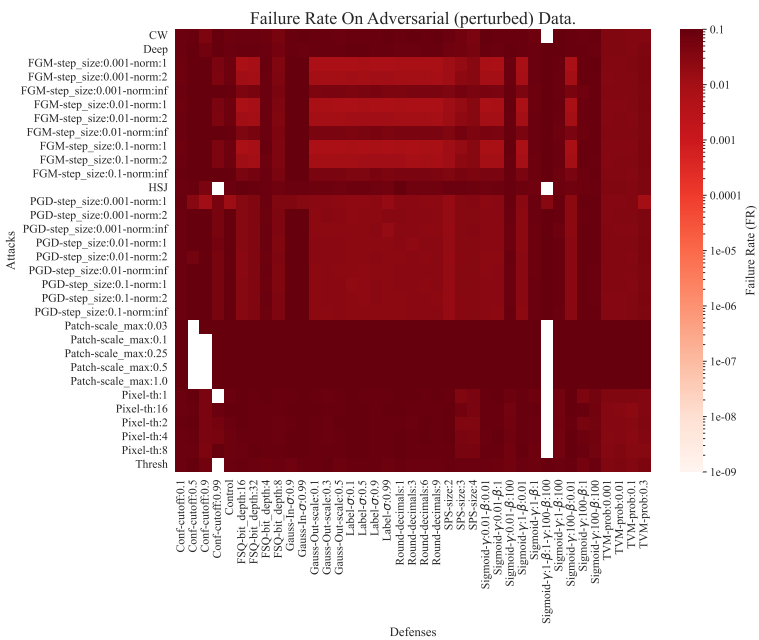


Fig. 15. The adversarial failure rate for all attacks and defenses for CIFAR-10. Darker red means worse performance and white indicates that no adversarial samples were found. This result is rather pessimistic, suggesting that all configurations fail to meet industrial standards.

increases. Furthermore, by simulating a black-box attack (HSJ attack), regularizing generated examples for false confidence (CW attack), finding minimal separating planes (Deep attack), and using advanced optimization techniques to generate highly confident false examples (Thresh attack), we were able to consistently fool the models in mere milliseconds per sample. The simplest attacks, which Madry et al. proposed in 2017, remain effective (PGD, FGM). Furthermore, proper step-size tuning seems to compensate for this simplicity. This is evidenced by right side of Figures 20 and 14, suggesting that the gradient ascent attacks might be a 'good enough' estimator of the generalized failure rate, especially given the astronomical gap between current test-set accuracy and regulatory requirements.

g) Percent Change in Accuracy: Figure 11 demonstrates how each attack fares against each defense. The color gradient starts at white, indicating the undefended model's performance against unperturbed data, becoming a more intense red as the accuracy decreases, with blue indicating an increase in accuracy compared to the control model. As we can plainly see, no single defense is able to consistently subvert an attacker. While some defenses do provide limited protection against more advanced techniques, their performance against gradient descent techniques is inconsistent at best. Furthermore, for DeepFool and HopSkipJump, we know that they could lead to even worse results for the defender, given a larger computational budget. While there is limited efficacy against gradient-based attacks for some defenses, advanced techniques like Adversarial Patch, DeepFool, and HopSkipJump consistently break models. That is, we provide substantial empirical evidence that the normally discussed accuracy from Equation 1 is, at best, an optimistic estimate of the real-world failure rate.

h) Percent Change in Failure Rate: Figure 16 depicts how each defense fares against each attack by comparing the change in failure rate when compared to the adversarial case on the undefended model. When we compare this plot to the one in Figure 11, we see that, in many cases, the defenses were able to decrease the adversarial failure rate when compared to the undefended model, which is what these defenses intended to do. However, increased safety (depicted as blue in Figure 16) is marginal at best—on the order of a few percent (see: Figure 4) when measured rates are many orders of magnitude from regulatory standards (see: Table I). Additionally, the potential downsides of a given defense are much larger (see: Figure 11). This marginal improvement in the failure rate is driven largely by the marginal time cost (see: Figure 13) rather than the accuracy (see: Figure 12). That is, the defenses increase the adversarial accuracy, but also take significantly longer (see: Figure 13).

i) MNIST vs. CIFAR-10: In addition to running these experiments on the CIFAR-10 dataset, we also ran them on the MNIST dataset using a simpler model provided by Madry *et al.* [55]. Figure 17 broadly verifies the behavior observed above on the CIFAR-10, wherein defenses tend to reduce model accuracy on the benign set, gradient-based attack efficacy is largely determined by hyperparameter tuning, and non-gradient-based attacks are still very effective. As with CIFAR-10, no defense was able to improve the benign (unperturbed) accuracy of the undefended model for every attack (see: Figure 18). Despite a smaller model and lower runtime requirements (compare Figure 19 to Figure 13), we see that training time is more-or-less the same. When we compare Figure 15 with Figure 21, we see that some of the defenses were much more effective at preventing gradient-descent attacks. However, they failed against the more computational expensive techniques of Patch, HSJ, and Deep. Then, when we examine failure rate (Figure 20), we see that despite being a different dataset and model architecture, we are still able to induce failures in only a few seconds, as with CIFAR-10. However, when we break this rate down into each attack and defense combination and measure a change in the failure rate (see: Figure 22), there seems to be a fairly consistent and marginal improvement driven by time (see: Figure 19) rather than accuracy (see: Figure 17), as with CIFAR-10. When we applied defenses to the simpler MNIST model and dataset (when compared to CIFAR-10), we found largely consistent results. However, we did find that there were more cases of failed adversarial attacks against MNIST, likely due to the significantly smaller dimensionality of both the dataset and model [32]. However, we would expect real-world data to be significantly higher resolution and full-color, unlike the black and white low-resolution images typical of MNIST, so those results probably underestimate the severity of the problem in real-world systems that use multiple multi-pixel RGB cameras to classify objects in real-time.

VIII. Limitations

A. True Failure Rate Estimation

While it is true that real-world noise can inadvertently become adversarial, it is obvious that not every possible noise vector will increase the loss for a given sample. We can, however, confidently say that the *true generalization error* lies between the test-set accuracy (Eq. 1) and the adversarial failure rate. Further work remains regarding the gap between these two estimates, but falls outside the scope of this paper. However, as we show in the results (Section VII) this fact hardly matters in the context of modern computer vision models, because **both of these measures** fail to meet safety-critical standards (see Table I) by many orders of magnitude (see Fig. 20 and Fig. 14.)

B. On optimal attacks

Additionally, while these attacks are quite efficient, none are provably the fastest possible attack. So, *at best*, they **underestimate** the failure rate. In the case of our safety-critical analysis, this **amplifies** and **does not diminish** our claims, raising serious concerns about using in-distribution test data as an indicator of real-world performance. Therefore, because these attacks are not provably optimal, this failure rate should not be taken as an absolute measure of the true failure rate. However, it is still a reliable metric for comparing the efficacy of two models as evidenced by the relatively consistent failure rates across various defenses and hyperparameter configurations for a given attack (right side of Figures 20 and 14).

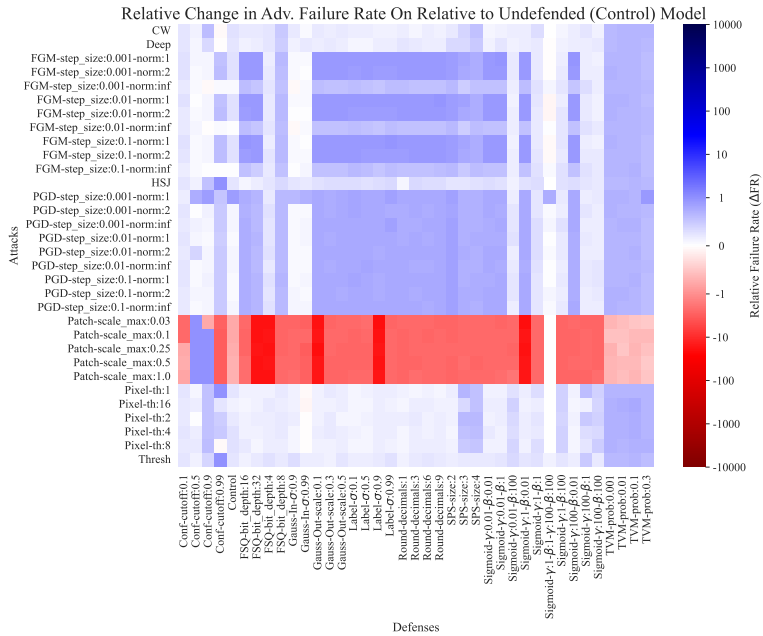


Fig. 16. The relative change in failure rate for the adversarial case, centered at the benign failure rate for CIFAR-10. Blue indicates an improvement relative to the benign (unperturbed) case. Red indicates a worsening of performance in the adversarial case compared to the benign case. White indicates no change. Note that the scale is logarithmic so that marginal gains and substantial losses can be seen clearly.

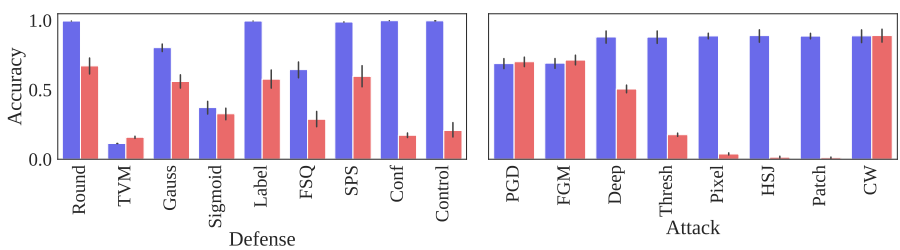


Fig. 17. The 95% confidence interval of adversarial and benign accuracies for each defense (left) and each attack (right) on the MNIST dataset, broken down by defense (left) and attack (right). One trial was conducted for each hyper-parameter combination.

C. Model Selection

In general, we did not choose the model architecture, but relied on two reference models provided by Madry et al. They have been tested and cited numerous times [55]. The point of this work is not to chase state-of-the-art results, but to evaluate robustness-maximizing techniques in a controlled manner while highlighting useful metrics and techniques for doing so. At the time of publishing, the authors are not aware of *any* architecture that meets safety critical standards when measured test-set accuracy sense (see Eq.1) and there's no demonstrated technique (to the knowledge of the authors) to reduce this in the adversarial sense without sacrificing accuracy or computational time in the benign case (see Eq.2, Fig. 20 and Fig. 14). So the general conclusion about the real-time safety-critical nature of modern neural networks would remain the same for any other architecture known to the authors.

D. On Attacker's Knowledge

In this paper, we have tested white-box attacks (FGM, PGD, CW), attacks that need access to the model probability outputs (Deep, Pixel, Thresh, Patch), and a single black-box method (HSJ), outlined in Sec. III. On one hand, it would seemingly be unfair to compare these methods under the same constraints. However, as we show in the results (Sec. VII), the apparent advantage of attackers with more knowledge of the model

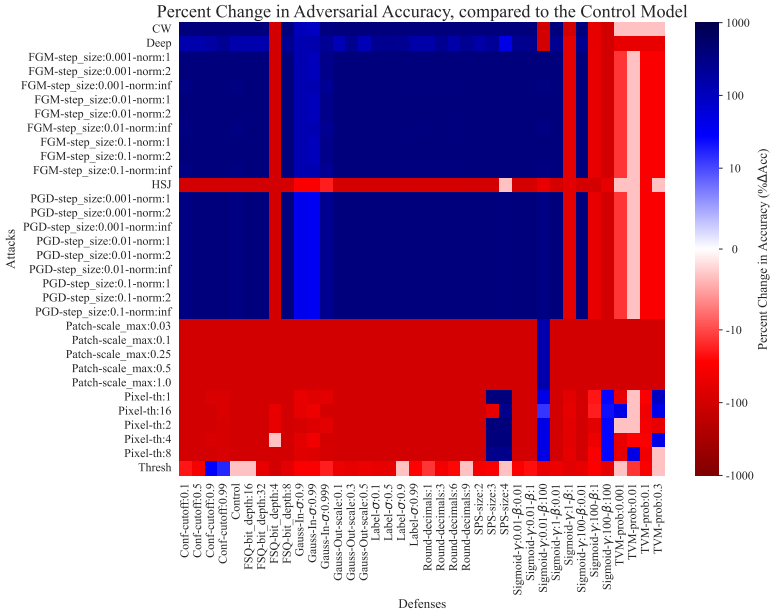


Fig. 18. The adversarial accuracy of each attack against each defense on the MNIST dataset. Blue would indicate an improvement relative to the undefended model. Red indicates that defense made a model worse. White indicates no change. This was computed using Equation 3.

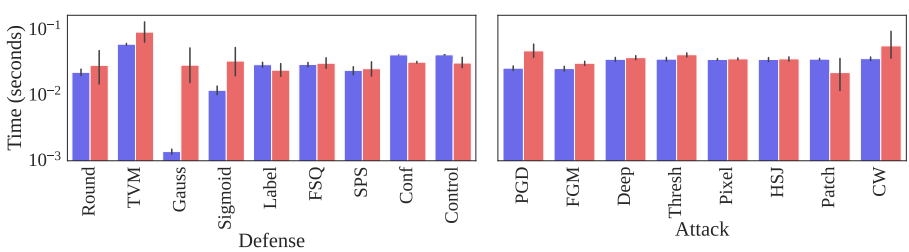


Fig. 19. The 95% confidence interval of prediction times (blue) and attack times (red) on the MNIST dataset, broken down by defense (left) and attack (right). Note that adversarial times are identical in each image since fitting and predicting are the same step in this case. One trial is depicted for each hyper-parameter combination.

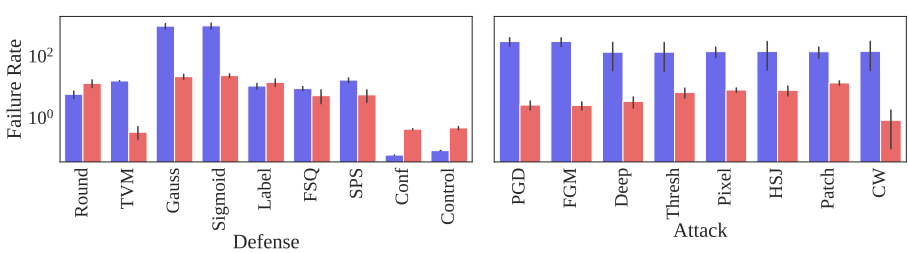


Fig. 20. The 95% confidence interval of failure rates, both benign (blue) and adversarial (red) on the MNIST dataset, broken down by defense (left) and attack (right). Note that no defense was able to improve upon the benign failure rate. This was computed using Equation 2.

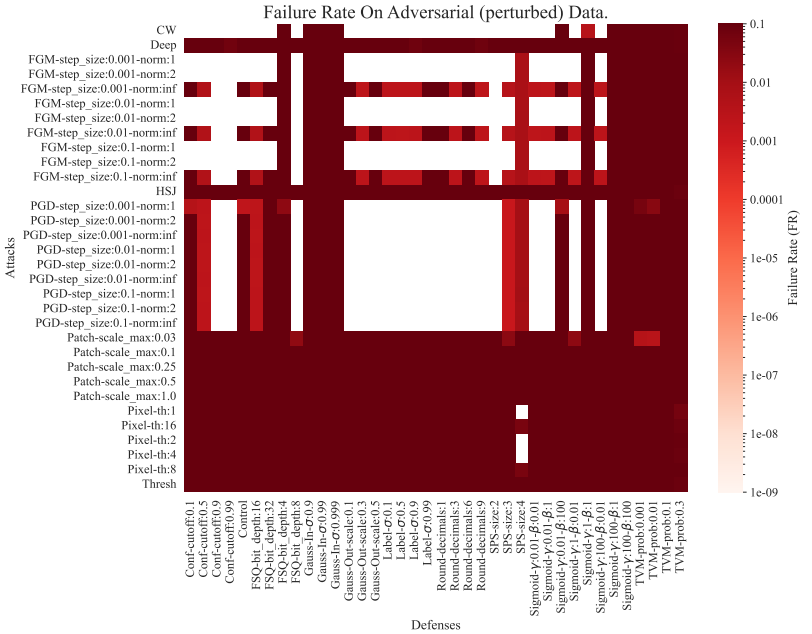


Fig. 21. The failure rate for the adversarial case on the MNIST dataset. A darker red indicates a worse performance and white indicates 0 induced failures. Note that the scale is logarithmic so that marginal gains and substantial losses can be seen clearly. White indicates cases where the attack was never successful.

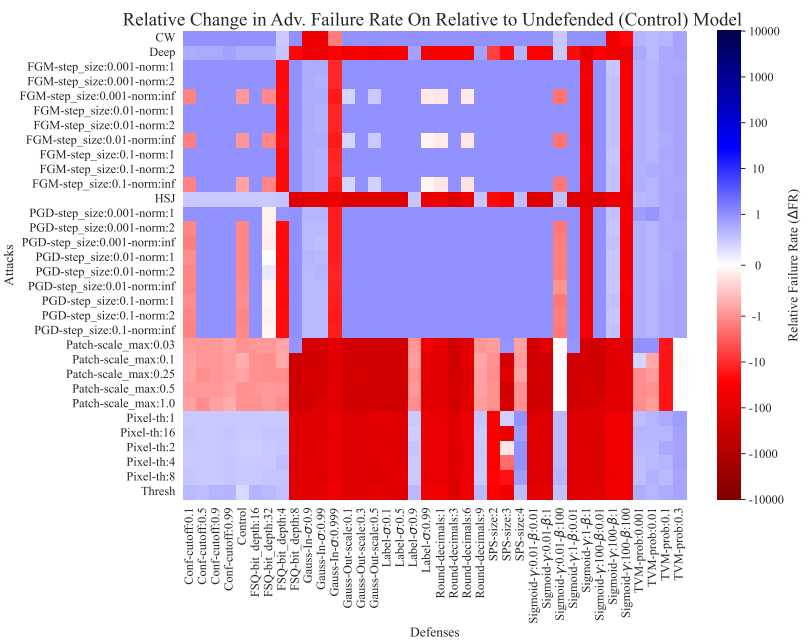


Fig. 22. The change in failure rate between the benign and adversarial cases on the MNIST dataset. Blue indicates an improvement relative to the undefended model. Red indicates that defense made a model worse. White indicates no change. Note that the scale is logarithmic so that marginal gains and substantial losses can be seen clearly. This was computed using Equation 4.

tends to disappear under the added computational burden. That is, the HSJ attack (Sec. III and Fig. 10), which only relies on hard class-labels and an offline model approximation, is consistently effective at fooling models while also outperforming attacks that have access to more information (see: Figs 3- 9). The interesting question, then, isn't necessarily how good of an adversarial sample one can generate, but *the rate* at which *any* misclassification can be induced. This *worst-case failure rate* will define the feasible upper bound on a target hardware architecture. Since these attacks vary substantially in runtime and information requirements, current methods relying on accuracy measures do not distill the efficacy of a given attack or defence from the perspective of any model-builder or attacker with a fixed computational budget. However, by controlling for the number of queries and normalizing by CPU-time (see: Section VI), we are able to isolate the effect of defence techniques against a wide-variety of idealized attackers that optimize for different distance metrics and are subject to very different constraints. By no means do we want to minimize the offline possibilities (see: Section III-C) of attacks like "Adversarial Patch" and "Hop Skip Jump". Instead, we seek to highlight the **computational triviality** of these attacks and raise sincere questions about the safety of these models in general.

IX. Conclusions

Neural networks are being deployed in a wide variety of industrial applications with real-world safety considerations, that despite high accuracy scores, fail against a wide variety of attacks that overwhelmingly require fewer computational resources than building the original model. While 'weak' attacks are fast, they require hyper-parameter tuning and retrospective evaluations that make them less effective, but nonetheless cheap-enough to execute, requiring only a few seconds of CPU time. The efficacy of even single-byte or single-pixel attacks against otherwise very accurate models raises questions not only about the intentional adversary, but also how a system will handle real-world, 'legitimate' anomalies like dust, lens aberration, and sensor failure.

When we consider the attack that finds the minimal class-separating distance, DeepFool, we see that 80% of all samples are corruptible under our meager one-byte distance constraint. When we remove this distance constraint, we find that nearly every sample can be fooled with the addition of an adversarial 'patch' on the original image (Figure 11) even when we constrain an attack to a small number of iterations ($\leq$ 10). That is, if we assume that our attacker has an unlimited budget, we cannot even hope to defend against these attacks. However, the relative ease of finding such adversarial examples suggests that these attacks provide an empirical estimate of a 'worst case' failure rate (see Equation 2) in their respective contexts. Therefore, adversarial model analysis provides a computationally cheap way to analyze the worst case failure rate of a system without having to collect, label, perturb, and predict many thousands of test images.

Furthermore, since this failure rate is based on process time, it is obvious that more powerful hardware, without underlying changes to the model architecture, would result in a larger failure rate since we merely fail on more samples in the same amount of time. That is to say, these problems are inherent to the model and not something that we can solve with more processor cycles. Furthermore, even when we obscure everything from the user except model output, we can consistently break models in as few as ten queries to a hard class-label API (*e.g.*, by using the HopSkipJump attack). We found that adding Gaussian noise to the model outputs or training images, reducing the bit-depth of the numerical calculations to match the precision of the image, and setting a confidence threshold were marginally effective defenses when compared to the control model. However, none of these defenses reach the regulatory standards required for safety-critical systems—in fact, they fail by several orders of magnitude. Furthermore, any gains seen in the failure rate (Figure 16) are inconsistent while universally reducing the accuracy on the original (unperturbed/benign) data (Figure 11). Improvements due to the defenses are rare and marginal, and they consistently make benign performance worse (see Figures 11 and 18). We find this to be true across both datasets and model architectures. Since we know that model accuracy scales with $\mathcal{O}(1/\sqrt{n})$ [79] and attacks seem to run in constant time (see Figures 13 and 19) for a fixed computational and query budgets of a few seconds and 1000 queries respectively.

Finally, due to the rate at which we can generate feasible misclassifications, we must question the real-world efficacy of any system that relies on these models to make predictions that meet the legal standards that define 'safe'. Despite this pessimism, adversarial model analysis proves to be a computationally efficient way to analyze and compare the out-of-distribution robustness of model architectures without the need to generate massive test sets from real world data.

X. Acknowledgements

Financial support has been provided in part by the Knut and Alice Wallenberg Foundation grant no. 2019.0352 and by the eSSENCE Programme under the Swedish Government's Strategic Research Initiative.

References

[1] PV Srinivas Acharyulu and P Seetharamaiah. A framework for safety automation of safety-critical systems operations. *Safety Science*, 77:133–142, 2015.

[2] Luis von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. Captcha: Using hard ai problems for security. In *International conference on the theory and applications of cryptographic techniques*, pages 294–311. Springer, 2003.

[3] Mohammed Al-Qizwini, Iman Barjasteh, Hothaifa Al-Qassab, and Hayder Radha. Deep learning algorithm for autonomous driving using GoogLeNet. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 89–96. IEEE, 2017.

[4] Ahamed Aljuhani. Machine learning approaches for combating distributed denial of service attacks in modern networking environments. *IEEE Access*, 9:42236–42264, 2021.

[5] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. *arXiv:1802.00420 [cs]*, July 2018.

[6] Victoria A Banks, Katherine L Plant, and Neville A Stanton. Driver error or designer error: Using the perceptual cycle model to explore the circumstances surrounding the fatal tesla crash on 7th may 2016. *Safety science*, 108:278–285, 2018.

[7] Julien Bect, Ling Li, and Emmanuel Vazquez. Bayesian subset simulation. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):762–786, January 2017.

[8] Guillermo Bernal, Sara Colombo, Mohammed Al Ai Baky, and Federico Casalegno. Safety++ designing IoT and wearable systems for industrial safety through a user centered design approach. In *Proceedings of the 10th International Conference on Pervasive Technologies Related to Assistive Environments*, pages 163–170, 2017.

[9] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion Attacks against Machine Learning at Test Time. *arXiv:1708.06131 [cs]*, 7908:387–402, 2013.

[10] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion Attacks against Machine Learning at Test Time. *arXiv:1708.06131 [cs]*, 7908:387–402, 2013.

[11] Battista Biggio, Giorgio Fumera, and Fabio Roli. Multiple Classifier Systems for Adversarial Classification Tasks. In Jón Atli Benediktsson, Josef Kittler, and Fabio Roli, editors, *Multiple Classifier Systems*, Lecture Notes in Computer Science, pages 132–141, Berlin, Heidelberg, 2009. Springer.

[12] Cara Bloom, Joshua Tan, Javed Ramjohn, and Lujo Bauer. Self-driving cars and data collection: Privacy perceptions of networked autonomous vehicles. In *Symposium on Usable Privacy and Security (SOUPS)*, 2017.

[13] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

[14] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv:1712.09665*, 2017.

[15] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR, 2018.

[16] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. *arXiv:1608.04644 [cs]*, March 2017.

[17] Hyunmin Chae, Chang Mook Kang, ByeoungDo Kim, Jaekyum Kim, Chung Choo Chung, and Jun Won Choi. Autonomous braking system via deep reinforcement learning. In *IEEE 20th International conference on intelligent transportation systems (ITSC)*, 2017.

[18] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv:1810.00069 [cs, stat]*, 2018.

[19] Antonin Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical imaging and vision*, 20(1):89–97, 2004.

[20] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. HopSkipJumpAttack: A query-efficient decision-based attack. In *IEEE symposium on security and privacy (sp)*, pages 1277–1294. IEEE, 2020.

[21] Li Chen, Jun Xiao, Pu Zou, and Haifeng Li. Lie to me: A soft threshold defense method for adversarial examples of remote sensing images. *IEEE Geoscience and Remote Sensing Letters*, pages 1–5, 2021.

[22] Travers Ching, Daniel S. Himmelstein, Brett K. Beaulieu-Jones, Alexandr A. Kalinin, Brian T. Do, Gregory P. Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M. Hoffman, Wei Xie, Gail L. Rosen, Benjamin J. Lengerich, Johnny Israeli, Jack Lanchantin, Stephen Woloszynek, Anne E. Carpenter, Avanti Shrikumar, Jinbo Xu, Evan M. Cofer, Christopher A. Lavender, Srinivas C. Turaga, Amr M. Alexandari, Zhiyong Lu, David J. Harris, Dave DeCaprio, Yanjun Qi, Anshul Kundaje, Yifan Peng, Laura K. Wiley, Marwin H. S. Segler, Simina M. Boca, S. Joshua Swamidass, Austin Huang, Anthony Gitter, and Casey S. Greene. Opportunities and obstacles for deep learning in biology and medicine. *Journal of the Royal Society Interface*, 15(141), 2017.

[23] Celia Cintas, Skyler Speakman, Victor Akinwande, William Ogallo, Komminist Weldemariam, Srihari Sridharan, and Edward McFowland. Detecting Adversarial Attacks via Subset Scanning of Autoencoder Activations and Reconstruction Error. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 876–882, Yokohama, Japan, July 2020.

[24] Matthew J. Colbrook, Vegard Antun, and Anders C. Hansen. Can stable and accurate neural networks be computed. *On the barriers of deep learning and Smale's 18th problem. arXiv*, 2101, 2021.

[25] International Electrotechnical Commission. *IEC 62304 Medical Device Software - Software Life Cycle Processes*. International Electrotechnical Commission, 2nd edition, 2006.

[26] International Electrotechnical Commission. *IEC 61508 Safety and Functional Safety*. International Electrotechnical Commission, 2nd edition, 2010.

[27] William A Corsaro. Something old and something new: The importance of prior ethnography in the collection and analysis of audiovisual data. *Sociological Methods & Research*, 11(2):145–166, 1982.

[28] Justin Cosentino, Federico Zaiter, Dan Pei, and Jun Zhu. The search for sparse, robust neural networks. *arXiv:1912.02386*, 2019.

[29] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *arXiv:2003.01690 [cs, stat]*, August 2020.

[30] Abbas Abou Daya, Mohammad A. Salahuddin, Noura Limam, and Raouf Boutaba. A Graph-Based Machine Learning Approach for Bot Detection. *arXiv:1902.08538 [cs]*, February 2019.

[31] Radosvet Desislavov, Fernando Martínez-Plumed, and José Hernández-Orallo. Compute and energy consumption trends in deep learning inference. *arXiv:2109.05472*, 2021.

[32] Elvis Dohmatob. Generalized No Free Lunch Theorem for Adversarial Robustness. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *PMLR*, 2019.

[33] Leonard Evans and Peter H Gerrish. Gender and age influence on fatality risk from the same physical impact determined using two-car crashes. *SAE transactions*, pages 1336–1341, 2001.

[34] Samuel G Finlayson, Hyung Won Chung, Isaac S Kohane, and Andrew L Beam. Adversarial attacks against medical deep learning systems. *arXiv:1804.05296*, 2018.

[35] The Organisation for Economic Co-operation and Development. OECD statistics. https://stats.oecd.org/ (visited 2022-04-20), 2020.

[36] National Highway Transportation Safety Administration's (NHTSA) National Center for Statistics and Analysis. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115 (visited 2022-04-20), 2015.

[37] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*, pages 1322–1333, Denver, Colorado, USA, 2015. ACM Press.

[38] Toshio Fukuda and Takanori Shibata. Theory and applications of neural networks for industrial control systems. *IEEE Transactions on industrial electronics*, 39(6):472–489, 1992.

[39] Judy Wawira Gichoya, Imon Banerjee, Ananth Reddy Bhimireddy, John L Burns, Leo Anthony Celi, Li-Ching Chen, Ramon Correa, Natalie Dullerud, Marzyeh Ghassemi, Shih-Cheng Huang, et al. Ai recognition of patient race in medical imaging: a modelling study. *The Lancet Digital Health*, 4(6):e406–e414, 2022.

[40] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv:1412.6572*, 2014.

[41] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[42] Fouad Hadj-Selem, Tommy Löfstedt, Elvis Dohmatob, Vincent Frouin, Mathieu Dubois, Vincent Guillemot, and Edouard Duchesnay. Continuation of Nesterov's smoothing for regression with structured sparsity in high-dimensional neuroimaging. *IEEE Transactions on Medical Imaging*, 37(11):2403–2413, 2018.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[44] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. *Complexity Theory and P vs NP*, chapter 4, pages 258–262. Springer, 2010.

[45] ICOH. Global estimates of occupational accidents and work-related illnesses 2017. *International Commission on Occupational Health'*, 2017.

[46] Daniel Jakubovitz and Raja Giryes. Improving dnn robustness to adversarial attacks using jacobian regularization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 514–529, 2018.

[47] Tong Jian, Zifeng Wang, Yanzhi Wang, Jennifer Dy, and Stratis Ioannidis. Pruning adversarially robust neural networks without adversarial examples. *arXiv:2210.04311*, 2022.

[48] Bernard Koch, Emily Denton, Alex Hanna, and Jacob G Foster. Reduced, reused and recycled: The life of a dataset in machine learning research. *arXiv preprint arXiv:2112.01716*, 2021.

[49] Shashank Kotyan and Danilo Vasconcellos Vargas. Adversarial robustness assessment: Why both l_0 and l_∞ attacks are necessary. *arXiv e-prints*, pages arXiv–1906, 2019.

[50] Hau Lam. New design-to-test software strategies accelerate time-to-market. In *IEEE/CPMT/SEMI 29th International Electronics Manufacturing Technology Symposium (IEEE Cat. No. 04CH37585)*, pages 140–143. IEEE, 2004.

[51] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672, 2019.

[52] Taesung Lee, Benjamin Edwards, Ian M. Molloy, and Dong Su. Defending against model stealing attacks using deceptive perturbations. *CoRR*, abs/1806.00054, 2018.

[53] Bo Li, Yevgeniy Vorobeychik, and Xinyun Chen. A General Retraining Framework for Scalable Adversarial Classification. *arXiv:1604.02606 [cs, stat]*, November 2016.

[54] Kaiji Lu, Piotr Mardziel, Fangjing Wu, Preetam Amancharla, and Anupam Datta. Gender bias in neural natural language processing. *Logic, Language, and Security: Essays Dedicated to Andre Scedrov on the Occasion of His 65th Birthday*, pages 189–202, 2020.

[55] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083*, 2017.

[56] Martin A Makary and Michael Daniel. Medical error—the third leading cause of death in the US. *BMJ*, 353, 2016.

[57] David J. Miller, Zhen Xiang, and George Kesidis. Adversarial Learning Targeting Deep Neural Network Classification: A Comprehensive Review of Defenses Against Attacks. *Proceedings of the IEEE*, 108(3):402–433, March 2020.

[58] Eric Monmasson, Lahoucine Idkhajine, Marcian N Cirstea, Imene Bahri, Alin Tisan, and Mohamed Wissem Naouar. Fpgas in industrial control applications. *IEEE Transactions on Industrial informatics*, 7(2):224–243, 2011.

[59] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[60] Blaine Alan Nelson. *Behavior of machine learning algorithms in adversarial environments*. University of California, Berkeley, 2010.

[61] International Standards Organization. ISO 26262-1:2011, road vehicles — functional safety. https://www.iso.org/standard/43464.html (visited 2022-04-20), 2018.

[62] Emre Pakdemirli. Artificial intelligence in radiology: friend or foe? where are we now and where are we heading? *Acta radiologica open*, 8(2), 2019.

[63] Jeman Park, DaeHun Nyang, and Aziz Mohaisen. Timing is almost everything: Realistic evaluation of the very short intermittent ddos attacks. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–10, 2018.

[64] Andrea Paudice, Luis Muñoz-González, Andras Gyorgy, and Emil C. Lupu. Detection of Adversarial Training Examples in Poisoning Attacks through Anomaly Detection. *arXiv:1802.03041 [cs, stat]*, February 2018.

[65] Ronald K Pearson. *Mining imperfect data: Dealing with contamination and incomplete records*. SIAM, 2005.

[66] Deborah Ramirez, Jack McDevitt, and Amy Farrell. *A resource guide on racial profiling data collection systems: Promising practices and lessons learned*. US Department of Justice, 2000.

[67] Leanna Rierson. *Developing safety-critical software: a practical guide for aviation software and DO-178C compliance*. CRC Press, 2017.

[68] Yuji Roh, Geon Heo, and Steven Euijong Whang. A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1328–1347, 2019.

[69] Andrew Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[70] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[71] Berkman Sahiner, Aria Pezeshk, Lubomir M Hadjiiski, Xiaosong Wang, Karen Drukker, Kenny H Cha, Ronald M Summers, and Maryellen L Giger. Deep learning in medical imaging and radiation therapy. *Medical physics*, 46(1):e1–e36, 2019.

[72] Vikash Sehwag, Shiqi Wang, Prateek Mittal, and Suman Jana. Towards compact and robust deep neural networks. *arXiv:1906.06110*, 2019.

[73] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[74] Mathieu Sinn, M Wistuba, B Buesser, MI Nicolae, and M Tran. Evolutionary search for adversarially robust neural networks. In *Safe Machine Learning workshop at ICLR*, 2019.

[75] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv:1704.03453*, 2017.

[76] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 601–618, 2016.

[77] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness May Be at Odds with Accuracy. *arXiv:1805.12152 [cs, stat]*, September 2019.

[78] Luu Anh Tuan, Man Chun Zheng, and Quan Thanh Tho. Modeling and verification of safety critical systems: A case study on pacemaker. In *2010 Fourth International Conference on Secure Software Integration and Reliability Improvement*, pages 23–32. IEEE, 2010.

[79] Vladimir Vapnik, Esther Levin, and Yann Le Cun. Measuring the vc-dimension of a learning machine. *Neural computation*, 6(5):851–876, 1994.

[80] Xianmin Wang, Jing Li, Xiaohui Kuang, Yu-an Tan, and Jin Li. The security of machine learning in an adversarial setting: A survey. *Journal of Parallel and Distributed Computing*, 130:12–23, August 2019.

[81] D. Warde-Farley and I. Goodfellow. Adversarial perturbations of deep neural networks. In D. Tarlow T. Hazan, G. Papandreou, editor, *Perturbations, Optimization, and Statistics*. The MIT Press, Cambridge, Massachusetts, 2017.

[82] Zihao Xiao, Xianfeng Gao, Chilin Fu, Yinpeng Dong, Wei Gao, Xiaolu Zhang, Jun Zhou, and Jun Zhu. Improving transferability of adversarial patches on face recognition with generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11845–11854, 2021.

[83] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv:1704.01155*, 2017.

[84] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, page 39–49, New York, NY, USA, 2017. Association for Computing Machinery.

[85] Yuchen Zhang and Percy Liang. Defending against whitebox adversarial attacks via randomized discretization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 684–693. PMLR, 2019.

[86] Billie J Zirger and Janet L Hartley. The effect of acceleration techniques on product development time. *IEEE Transactions on Engineering Management*, 43(2):143–152, 1996.

Paper II

Massively parallel evasion attacks and the pitfalls of adversarial retraining.

Charles Meyers, Tommy Löfstedt, Erik Elmroth

EAI Endorsed Transactions on Internet of Things, Volume 10, 2024

Can we overcome an adversary by generating or using more data?

EAI Endorsed Transactions Preprint Research Article/Editorial

Massively Parallel Evasion Attacks and the Pitfalls of Adversarial Retraining

Charles Meyers[1], Tommy Löfstedt[1] Erik Elmroth[1]

[1]Department of Computing Science, Umeå University, Umeå, Sweden

Abstract

Even with widespread adoption of automated anomaly detection in safety-critical areas, both classical and advanced machine learning models are susceptible to first-order evasion attacks that fool models at run-time (e.g. an automated firewall or an anti-virus application). Kernelized support vector machines (KSVMs) are an especially useful model because they combine a complex geometry with low run-time requirements (e.g. when compared to neural networks), acting as a run-time lower bound when compared to contemporary models (e.g. deep neural networks), to provide a cost-efficient way to measure model and attack run-time costs. To properly measure and combat adversaries, we propose a massively parallel projected gradient descent (PGD) evasion attack framework. Through theoretical examinations and experiments carried out using linearly-separable Gaussian normal data, we present (i) a massively parallel naive attack, we show that adversarial retraining is unlikely to be an effective means to combat an attacker even on linearly separable datasets, (ii) a cost effective way of evaluating models defences and attacks, and an extensible code base for doing so, (iii) an inverse relationship between adversarial robustness and benign accuracy, (iv) the lack of a general relationship between attack time and efficacy, and (v) that adversarial retraining increases compute time exponentially while failing to reliably prevent highly-confident false classifications.

Received on 9-2023; accepted on 10-2023; published on 7-2024

Keywords: Machine Learning, Support Vector Machines, Trustworthy AI, Anomaly Detection, AI for Cybersecurity, *EAI Endorsed Transactions*

doi:10.4108/eetiot.6652

1. Introduction

There are several types of attacks that target machine learning models or data at different phases. Below we examine attacks where an attacker seeks to *evade* a classifier by confusing the model at run-time (e.g. a penetrated network or a malicious application being detected as benign). Robustness—the ability to withstand attacks—is usually measured by the accuracy gap before and after being attacked (benign and adversarial accuracy, respectively). While much work has gone into evaluating the robustness for neural networks [1–3] and regression algorithms [4], Machine learning using support vector machines have proven to be useful in compute constrained applications like system intrusion detection [5], network anomaly detection [6], and image recognition [7].

Researchers cite concerns about evaluating against weak attacks [8, 9], the hard problem of step-size optimization [10], the weakness of defense generalization [11], and the transferability of attacks [12], which contributes to a body of work that consistently fails to be reproducible [3] while relying on increasingly massive computational resources for increasingly marginal gains [13]. In short, there is a strong need for generalized testing against strong adversaries [9].

Since a theory of robust generalization remains evasive, it is necessary to evaluate the robustness of defenses across the broadest-possible number of hyperparameters with the understanding that they are drawn from a continuous and infinite space. Without a strong theory of generalization, currently, the only way to evaluate attacks is through brute force. Because this process is computationally expensive for large hyper-parameter sets, we propose a scalable framework for

*Corresponding author. Email: cmeyers@cs.umu.se

attack generation, useful both for defense evaluation and distributed adversarial attacks.

1.1. Contributions

We present a novel parallel attack generation framework allows for massively generating adversarial examples across multiple cores or multiple machines, of particular use in scenarios that demand exploring a large hyperparameter space.

- We provide an extensible and scalable code base for finding optimal attack configurations for a variety of attacks and defences.
- We provide a method for faster attack generation for both benign (re-training) and adversarial circumstances, with easy extensibility to a variety of algorithms with large hyperparameter spaces including models, defences, and attacks.
- We show that attack efficacy is uncorrelated with attack time and more dependent on the total perturbation distance and the step size at each iteration.
- We show that, the relationship between step size, perturbation size, and false confidence is highly complex, has a large hyperparameter space, and a product of both the model and the data set and that characterizing the feasible space is critical to generalized robustness.
- Using a strong adversary as determined by massively parallel tests, we found that adversarial retraining was impractical in terms of both compute time and benign model accuracy against a strong adversary on a variety of binary datasets.

2. Related Work

Prior research has shown the inverse relationship between model robustness and model accuracy empirically [10, 14]. Tsipras et al. [14] highlighted this trade-off while examining neural nets on several image data-sets. Other experiments have shown a strong inverse relationship between model robustness and model accuracy for a large number of samples and accurate models [15] more generally. Raghunathan et al. [15] offer a theoretical explanation, suggesting that this trade-off is an artifact of imperfect sampling. Even after explicitly minimizing the gap between benign and adversarial accuracy, the adversarial model had a nearly 6 times increase in adversarial error relative to the benign case. In addition, it was recently proven that *all* classifiers are vulnerable to attacks from an adversary [16], which raises issues for safety-critical and real-time systems. Since all classifiers are doomed to fail against such attacks, then, at the very least, it is critical to quantify classification robustness.

Current research suggests adopting only strong attacks in these evaluations [9], but the strength of an attack is unknown prior to evaluation, and is context-dependent. Furthermore, Croce et al. [3] showed fifty cases where modern, published research failed to have reproducible robustness. Other research has shown that randomized smoothing, obfuscated gradients, and even non-differentiable models fail to produce strong defenses against the proposed attacker [8, 17, 18]. This is intuitive—creating a boundary condition sensitive to a particular attack does not remove the existence of a new gradient to be exploited.

3. Background

In this section, we briefly discuss our choice in models, attacks, and defences.

3.1. Support Vector Machines

Support vector machines [19] can be used for both classification and regression. The kernelized versions include arbitrary data transformations (through kernels) that casts a data-set into a higher-dimensional space, where the classes are linearly separable. The resulting models are determined by solving convex optimization problems, as detailed below. Trafalis et al. [20] show that under benign perturbations, these models are robust and numerically stable, but little is known about their robustness against an adversarial attacker.

A support vector machine is trained by solving the Lagrange dual problem [19],

$$\max_{c_1,\ldots,c_n} \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i \langle x_i, x_j\rangle y_j c_j \qquad (1)$$

$$\text{subject to } \sum_{i=1}^{n} c_i y_i = 0 \quad \text{and} \quad 0 \le c_i \le \frac{1}{2n\lambda}, \forall i,$$

where $x_1, \ldots, x_n$ is the set of training examples, $y_1, \ldots, y_n$ are the training labels for the n samples, the c_i are the dual variables found during training, and λ is a regularization parameter that controls the complexity of the decision boundary by penalizing wrong classifications. Solving this quadratic problem requires at least $O(n^2)$ dot products, making it computationally expensive for large data-sets.

In addition to the non-kernelized linear model above, we also evaluated the kernelized version of this model for transformed features, $\phi(x_i)$. Inner products of the transformed features can be computed using the kernel trick [19],

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j)\rangle,$$

which gives a closed form expression of the inner product of the transformed features. New data points, x, are predicted using

$$\hat{y} = \sum_{i=1}^{n} c_i y_i K(x_i, x),$$

where the c_i, for $i = 1, \ldots, n$ are obtained by solving the maximization problem in Equation 1, but using a kernel function,

$$\sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i c_i K(x_i, x_j) y_j c_j.$$

The nominative support vectors are denoted

$$S = \left\{ c_i \,\middle|\, 0 < c_i < \frac{1}{2n\lambda}, i = 1, \ldots, n \right\}.$$

Computing the values of c_i in this set is equivalent to inverting a matrix naively, which has complexity $O(|S|^3)$ [21], where $|S|$ is the number of support vectors for which $0 < c_i < \frac{1}{2n\lambda}$. However, the actual run-time varies wildly between kernel choice, data-set, and parameter choice. Merely verifying that a vector c_i is a solution to the quadratic programming problem requires computations that scale with the number of support vectors, $|S|$, and the number of samples, n, giving a complexity of $O(n|S|)$ [6]. Furthermore, with a non-zero error rate, E, it has been shown [22] that $|S|$ is asymptotically equivalent to $2nE$, giving a complexity of

$$O(pn^2E).$$

The kernelized version has a complexity associated with the cost, κ, that varies significantly with the kernel and has the same as the non-kernelized version such that

$$O(pn^2E) + O(\kappa).$$

This kernelization step can itself be quite costly in terms of compute time and it's complexity is examined below.

Radial Basis Function Kernel. Kernel functions can be used to cast a problem from a feature space of p dimensions into an infinite-dimensional space using the kernel function as a similarity measure between each pair of samples. The radial basis function kernel is given by

$$K(x_i, x_j) = e^{-\gamma d(x_i, x_j)},$$

where $\gamma = (2\sigma^2)^{-1}$, and d is any suitable distance metric. The gradient is given by

$$\nabla K(x_i, x_j) = -\gamma e^{-\gamma d(x_i, x_j)} \nabla d(x_i, x_j),$$

with a distance metric that scales with p (e.g., an inner product), and with n^2 such computations, the total complexity becomes $O(pn^2)$.

Polynomial Kernel. The polynomial kernel is given by

$$K(x_i, x_j) = \Big(\langle x_i, x_j \rangle + r \Big)^D,$$

where r is a tune-able parameter and D describes the degree of the polynomial (also tune-able). Its gradient is given by

$$\nabla K(x_i, x_j) = \Delta D (\langle x_i, x_j \rangle + r)^{D-1},$$

where Δ is a diagonal matrix with the inputs, $x_{i,1}, \ldots, x_{i,p}$, $x_{j,1}, \ldots, x_{j,p}$, on the diagonal. Computing the gradient means an inner product that scales with p (the dimension of x_i), D multiplications of the inner product value, and n^2 such computations, giving a total complexity of $O(pDn^2)$.

The lower bound of complexity is given by the number of dot products in the Lagrangian in Equation 1, which has an overall complexity of $O(pn^2)$, showing that the attack time will be dominated by the number of attacked samples rather than kernel choice.

SVMs: cost-effective attack and defence analysis. We demonstrate the experiments using kernelized SVMs, but note that the metrics and analyses generalize to more elaborate models. Firstly, we focus on the training-time complexity of these models, which is already known to be significantly smaller than popular neural network architectures [23]. Secondly, our goal is not to chase state-of-the-art results, but to unequivocally demonstrate that retraining methods (dictated by NIST [24]) increase the accuracy against a given set of adversarial attacks at the cost of confidence in the unperturbed case will be unlikely to meet the risk-reduction standards outlined in ISO 26262 [25]. By using SVMs instead of more costly models, we were able to evaluate a larger number of hyperparameters on a fixed computational budget, a necessity for large and complex hyperparameters spaces. Furthermore, since the generated data was defined to be linearly separable in a p-dimensional space, any and all of these models *should* work quite well.

3.2. Projected Gradient Descent Attacks

Projected gradient descent (PGD) has become a standard measure for model robustness. Carlini and Wagner [17] proved that any boundary condition can be shifted by a relatively small number of points by optimizing under two constraints—one that maximizes the classification error, and one that minimizes the adversarial perturbation. This 'fast-gradient' attack was extended with PGD, a 'universal' first-order white-box attack and numerical optimization algorithm [2]. It has since become the standard way to measure robustness of a particular model or defense due to its universality. The iteration scheme is

$$x^{(k+1)} = P\Big(x^{(k)} + \eta^{(k)} \nabla f(x^{(k)}) \Big),$$

where $x^{(k)}$ is the adversarial example at iteration $k = 1, \ldots, N$, the N is the number of iterations, $P(x)$ is a projection of x onto a convex set (e.g., a norm ball in the feature space) with radius d_{max} (chosen by the experimenter), $\eta^{(k)}$ is the step size in each iteration, and $f(x^{(k)})$ is the original model function at iteration k. An attack thus takes ascent steps in the direction of the loss gradient, attempting to maximize the loss of the attacked model, with the projection step used as a means to adhere to any other constraints.

3.3. On Attack Choice

Most importantly, we note that *any* attacks that rely on this kind of secondary optimization of a large hyperparameter space (see Figure 3 and references [3, 17, 26–34]) will necessarily run no faster than a naive implementation in parallel, particularly if we tune the batch-size, step-size, and iterations to minimize the run-time and maximize the loss for some ideal by using a hyperband [35] or multi-objective search [36, 37]. While those methods are outside the scope of this paper, the source code we provide, allows for such searches on many different frameworks, defences, and attacks by merely changing the contents of a single configuration file[1], meaning that finding an *optimal* attack for any model using frameworks such as Scikitlearn, Tensorflow, Pytorch, and MXNET would be trivial.

3.4. Adversarial Retraining

Adversarial retraining inherits the time complexity of both the model and the attack above such that the complexity is

$$O(n^2 p),$$

before actually creating the retrained model, which has $2n$ samples, giving us the complexity

$$O(n^2 p) + O\left((2n)^2\right) = O(n^2 p)$$

which will add significant training time to the model (already measured in hours or days) with the unintended side effect of reducing benign accuracy [11].

Adversarial retraining has been proposed as a general solution to this problem [10, 11]. In the naive version, the final iterate from the PGD, x^*, is appended to the training set, labelled as malicious, and the training and attack cycle is repeated until accuracy converges or no novel samples can be generated [10].

[1] https://hydra.cc/docs/plugins/optuna_sweeper

4. Massively Parallel Evasion Attacks

Generating parallel attacks allows for a larger robustness evaluation while simultaneously allowing for a faster generation of adversarial examples for adversarial re-training. The massively parallel attack generation framework is shown in Figure 1.

Given that model attack parameters are drawn from an infinite space and that published model robustness tends to be over-reported [3], we propose a massively parallel attack generation framework that reliably evaluates a much larger set of attacks than is common practice. Furthermore, attacks should be examined not only in the context of raw accuracy numbers, but also their ability to prevent highly-confident false classifications as well as the time needed to break a model.

Attacker's Goal: We consider a classification algorithm $f : X \to Y$ for samples in some feature space $x \in X$ to a label in the set of classes $y \in Y = \{-1, 1\}$ where 1 represents the malicious class. Since the estimator returns a probability in $[0, 1]$ for each one-hot label [38], we assign the label with the highest probability for evaluation purposes. Our attacker's goal is to shift the classification of at least one input example such that the confidence of a false classification is $> 99\%$. The feasibility of such attacks is examined below.

Attacker's Capability: In the case of evasion attacks, the adversary can only modify data at test time. Prior attacks have allowed arbitrary and significant change to the original feature space. However, this is not feasible in many real-world scenarios [39]. We also assume that the attacker is relatively resource-constrained, ruling out attacks that require specialized hardware (like deep-learning). A more likely massive attack scenario involves a malicious advertisement [40] or an insecure network-connected, low-power device [41]. We also assume a single model input/output stream shared among all attacks which reduces the detection surface relative to attacks probing the model separately. To meet this goal, we supplied 100 samples to the attacker, but, as we show below (Figure 5c), attacks can be successful when supplied with only a single example. Despite this constraint, the attacker can still reliably generate false classifications.

4.1. Attacker's Knowledge

In order to measure robustness of a given model, it is assumed that the attacker knows most things about the model, including the distribution, shape, and feature space of the training set; the type of model used and it's parameter space; the gradients with respect to the optimization criteria; and feedback from the model in

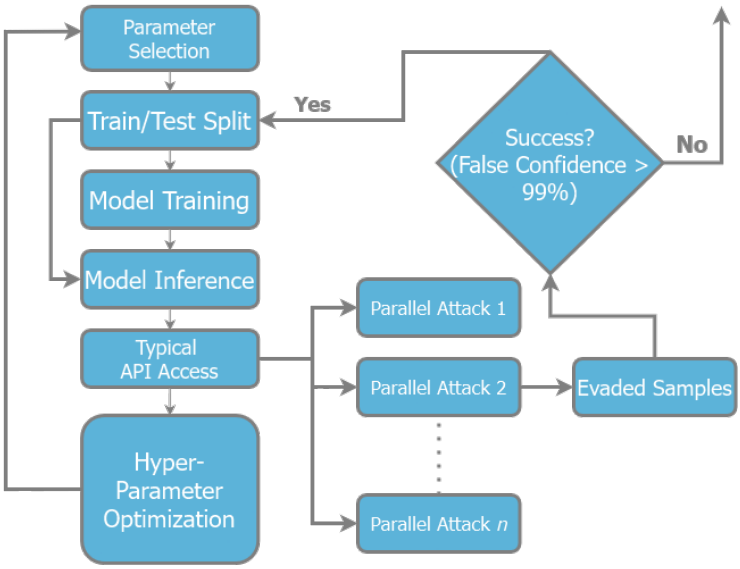


Figure 1. Massively Parallel Attack and Re-training Framework. This depicts our experimental pipeline wherein we build KSVMs with various kernels, run several attacks in parallel, and then evaluate the attacked samples on the models. Optionally, we collect highly confident false examples for adversarial retraining.

the form of model probability output. While it may seem like a prohibitively large set of assumptions, we outline possible attack vectors below. In our case, the attacker queries the model with an adversarial sample and is given the ℓ_∞ norm which returns the largest deviation of a single feature for a given sample rather than the more granular information provided by other standard distance metrics, like the ℓ_2 or ℓ_1 norms. It also ensures that no single feature is perturbed by more than d_{max}. As an added benefit, it marginally reduces run-time and memory requirements, with the savings scaling with the number of features. Below we examine both the ideal and realistic scenarios for these attacks.

Perfect Knowledge: While assumed the adversary has access to the model gradients with respect to the loss function, it can be approximated through Monte Carlo methods or via other attacks [42, 43]. It is not necessary to know all of the model parameters, just the weights and biases that compose the fitted model. Although even this constraint is broken by other attacks [42, 43]. The adversary can transform sample data, but must remain within a maximum distance d_{max} for each feature. For our purposes, we chose this distance to be one standard deviation for a given feature, ensuring transformed data does not stray too far from the benign data and decrease the separability for the retrained classifier. Other works [10, 11, 44] try to minimize the requisite perturbation distance, but because we are dealing with numeric data and not image data, data that falls within the the first standard deviation would likely not look adversarial to a human observer. The same cannot necessarily be said for image data in which it is natural to have highly variant data and a large contrast between different regions. In many cases, perfect knowledge is provided normally by the peer-review process and published model weights. However, many models are proprietary and can only be accessed through an API that returns only the classification, either as a probability distribution or the $argmax$ of that distribution [45].

Limited Knowledge: Even though our attack scenario only includes perfect knowledge, prior research [39, 42, 43, 46, 47] has shown that a surrogate model and data-set can be used to approximate $f(x)$ by $\hat{f}(x)$ and build a model using the class labels provided by the attacked model at test-time. Tramèr et al. [45] examined popular machine learning as a service platforms that return confidence values as well as class labels, showing that an attacker can build a proxy model by querying $p+1$ random p–dimensional inputs for unknown $p+1$ parameters. Further researchers [46] were able to

reverse engineer the training data-set through black-box attacks against a model that returns confidence levels, with the caveat that the inferred data might be a meta-prototypical example that does not appear in the original data-set. Fortunately for our attacker, such examples are still useful for determining the underlying data distributions even if they manage to preserve some of the privacy of the original data-set. Shokri et al. [48] presented a membership inference attack that determines whether a given data point belongs to the same distribution as the original training data using a set of proxy models. Although we tested only the perfect knowledge scenario, there are myriad ways for an attacker to get access to otherwise private data using nothing but standard machine learning APIs.

4.2. Attack Generation Algorithm

Under the above assumptions, the optimal attack strategy seeks to find a set of feature values for a sample, $x^{(0)}$, such that $x^* = \arg\min_x \hat{f}(x)$ and $d(x^*, x^{(0)}) \leq d_{max} = 1$ since we centered and scaled the data to ensure each feature had the same variance ($\sigma^2 = 1$). The algorithm is outlined in Algorithm 1.

Algorithm 1: Parallel PGD

Input: A set of step sizes $\{\eta\}$; $\{d_{max} > 0\}$, a set of maximum perturbation constants; a set of batch sizes, $\{m\}$; a trained model $f(x)$; $X = \{(\bar{x}^{(0)}, \bar{y})\}$ a set of unperturbed samples and their corresponding labels; $\{I\}$, a set of maximum iterations; and a projection operator

$$P_{d_{max}}(X) = \left\{ \text{argmin}_{x^*,\, d(x^*,x) \leq d_{max}} \|x^* - x\| \right\}_{x \in X}$$

Output: x^*, a sample with perturbation no greater than d_{max}

Generate a grid to search over from the supplied parameters:

$$G = \text{AllCombinations}\Big(\{\eta\}, \{m\}, \{I\}, \{d_{max}\}\Big)$$

foreach $(\eta, m, I, d_{max}) = g \in G$ **in parallel do**
  $i \leftarrow 0$
  while $i \leq I$ **do**
    foreach $X_m^{(i)} \subseteq X$ **do**
      $X_m^{(i+1)} \leftarrow P_{d_{max}}\Big(X_m^{(i)} + \eta \nabla f\Big(X_m^{(i)}\Big)\Big)$
    end
    $i \leftarrow i + 1$
  end
end

4.3. Attack Complexity

If we assume perfect parallelism in the outermost while loop (in Algorithm 1) under the possible attack scenarios outlined above, then our attack complexity scales with the number of iterations, I, the number of batches, b, and the number of samples per batch, n_{batch}. With $m = n_{batch} \cdot b$, this gives us a complexity of

$$O(I \cdot m).$$

Our own experiments (Figure 3) show that iterations do little to change attack efficacy in themselves. So, if we assume that $N \ll m$, this model scales linearly with the number of perturbed samples, giving a fundamental advantage over the model which is trained in polynomial time. Furthermore, this m can be several orders of magnitude smaller than the training database size n, with successful attacks occurring even when a single data point is supplied to the attack at a time (see Figure 5c). So, it's possible that a 'good' attack can operate in linear time. Figures 2a and 5c confirm the existence of such attacks. If we assume that the API can correctly identify and mitigate some adversarial queries with some error rate, $E \in [0, 1)$, then the actual number of real-world API queries, Q, needed by an attacker would be

$$Q = Im(1 - E).$$

That is to say, as the error rate increases, an attack becomes easier in real world circumstances. This is a particular detriment to the model builder who relies on adversarial retraining (see: Fig. 5d).

5. Evaluations

Our experimental methods are outlined in detail below.

5.1. Data-set

To show that these problems hold for 'nice' data, we generated many numeric datasets. We sampled n Gaussian distributed points near opposing corners of a hybercube in p dimensions, separated by an ℓ_2 distance of ten. We generated twenty unique datasets with the combinations of $p \in [10, 10^2, 10^3, 10^4]$ and $n \in [10^2, 10^3, 10^4, 10^5, 10^6]$. We also ran the framework on the the intrusion-detection KDD-NSL dataset[49], selected in such a way as to avoid duplicate rows, a common critique of the original [50] as well as the Truthseeker dataset[51] that divides malicious and benign twitter users based on a variety of usage data (see: Appendices A and B). For adversarial retraining, the positive label was used for new data, classifying it as 'malicious' and of the same class as a variety of network-based attacks included in the original data set. For our evaluations, we used 100,000 training samples, and one hundred consistent samples in the test set.

5.2. Experimental Setup

In our parallel implementation, we dedicated one core to each attack and tested a large number of hyperparameters at the same time using 'joblib'[2]. We used the `optuna` [52] framework for handling scheduling[3], hydra[4] for hyperaparamater configuration management, and dvc[5] to track results and guarantee reproducibility. We also provide source code[6], designed to be extensible to other machine learning frameworks (*e.g.*, Keras, Tensorflow, Pytorch, MXnet, etc.), defences, and attacks while scaling to diverse and distributed systems. In addition to running the model selection in parallel, we split the attack parameter space to run in parallel, each attack operating on the same set of test data. For our experiments, we used a 2.15Ghz AMD EPYC 7702P processor with 128 cores. We used `scikit-learn`[7] and `libsvm` [53] to build the models and IBM's Adversarial Robustness Toolbox (`art`)[8] to generate attacks. Although we restricted our tests to a single machine to make the time-complexity analysis more straight-forward, `optuna` is capable of scaling to multiple machines in a cluster. The scheduler spends around a hundred microseconds on every task, but this is negligible compared to the training times on a reasonably sized database and comparable to a well-chosen attack (Figure 2a). Although we parallelized model fitting and attack creation in the same way, our parallel attack paradigm means that each attack time was measured individually while model building times were measured as a whole and normalized by the number of tested models since increasing the model hyper-parameter search space will obviously increase the run-time requirements. In this way, we attempt to compare the average model building time for a given set of parameters with a single attack.

For model building purposes, we evaluated every order of magnitude in $[10^{-5}, 10^{5}]$ for both c and λ (Equation 1) for each of the linear, polynomial, and RBF kernels. We also tested balanced class weight and naive class weight as well as one vs. one and one vs. rest classifiers for each kernel. For the polynomial kernel, we evaluated degrees $D \in \{1, 2, 3, 4, 5\}$ in addition to the parameters above. Because SVMs require a large hyper-parameter search, we parallelized the search and normalized the reported time for each kernel by the cardinality of the grid search. Reported times are the average model fitting wall time on a single core. Attack times are reported as wall time per attack. We examined the attack efficacy in the case of perfect knowledge as outlined above.

When controlling for the training set size, we evaluated the number of samples for several multiples of ten in $[10, 10^{6}]$, with the largest model being used for all subsequent experiments. In addition to tracking the attack time for the entire attack space. For the attack phase, we tested maximum perturbations in $\{0.001, 0.01, 0.1, 0.2, 0.3, 0.5, 0.7, 1.0\}$ but varied the step size for each power of ten in $[10^{-4}, 1]$. We tested iterations in $\{1, 10, 10^{2}, 10^{3}\}$ and batch sizes in $\{1, 10, 10^{2}, 10^{3}\}$. For all tests but AT, we withheld 1000 benign samples to test the models and to generate the attacks. For AT, we reduced the size of the training database to ten-thousand (from one-hundred thousand) and evaluated the adversarial and benign accuracies on 1000 samples to make the AT process computationally feasible. We also examined the efficacy of AT and its ability to defend against a new set of attacks when applied to all three kernels.

6. Results and Discussion

In the section below, we examine the performance, the robustness, the attack time, the efficacy of attack hyperparameter tuning, and the pitfalls of adversarial retraining.

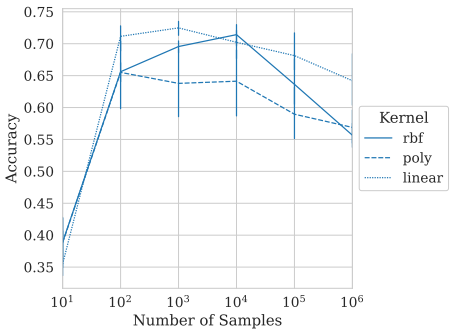
6.1. Performance:

By using a massively parallel optuna [52] implementation, we were able to generate tens of thousands of strong examples from one-thousand input-output pairs in a way that extends to other attacks, frameworks (*e.g.* MXNet, Pytorch, Keras, Tensorflow), and defences (*e.g.* ART). Our implementation[9] allocates one core per process and runs them in series if there are more processes queued than available cores. All generated models, data, and results are stored to disk, as well as in an sqlite database, all specified in a single configuration file, allowing for arbitrary divisions of the evaluation pipeline across any number of diverse and distributed machines. As we see from the experiment (Figure 2c), the lower bound of these calculations is a few hundred milliseconds, given that is how long they take when executed on a single core in series, so the scheduler overhead appears to be minor even though it's statistically significant.

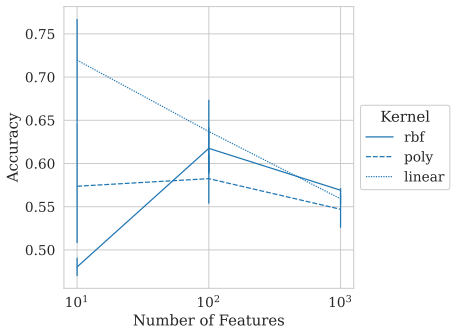
6.2. Accuracy and Robustness:

Much research has been devoted to the apparent trade-off between robustness and benign accuracy (see

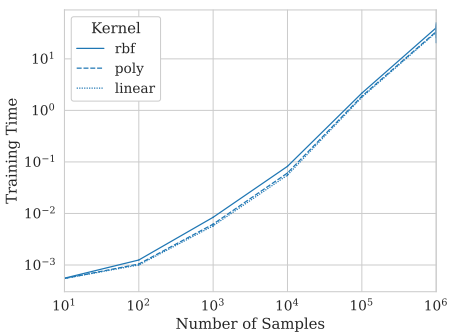
[2] https://joblib.readthedocs.io
[3] https://optuna.readthedocs.io
[4] https://hydra.cc
[5] https://dvc.org
[6] Our repository
[7] https://scikit-learn.org
[8] https://adversarial-robustness-toolbox.readthedocs.io
[9] Our Github Repository



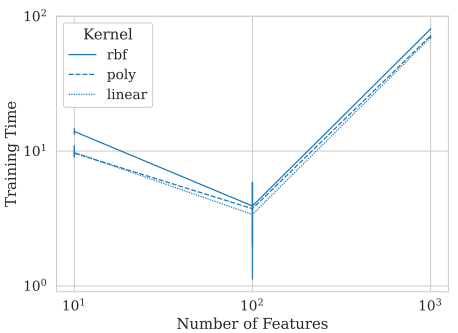
(a) Model Performance vs Database Size: This depicts the benign performance of a model (*e.g.* accuracy on unperturbed data) when trained on databases of difference sizes for each tested kernel.



(b) Model Performance vs Feature Space: This depicts the benign performance of a model (*e.g.* accuracy on unperturbed data) when trained on a differing number of features. In addition, marginal features are less correlated with the label than earlier features, simulating the addition of a large number of noisy features, leading to increasingly inaccurate models.



(c) Training and Attack Times vs Database Size: This shows the time requirements to build models and attacks on databases of different sizes.



(d) Training and Attack Times vs Feature Space Size: This shows the time requirements to build models and attacks on feature space of different sizes.

Figure 2. This depicts the benign accuracy (top) and training times (bottom) across all three kernels, varying the number of samples (left) and the number of features (right). The bars reflect the 95% confidence interval for all tested configurations.

Related Work) and we see signs of it across all of our experiments. The second experiment (Figure 2a) confirmed an inverse relationship between robustness and model accuracy. Even when AT is able to perfectly classify adversarial examples in the new training set (Figure 5d), average error in the benign circumstance increased from 0.02% to roughly 50%. The adversarial accuracy against strong attacks increased, but at a substantial cost to adversarial accuracy. This would be catastrophic in safety- or security-critical settings. In Figure 2a we can see that model accuracy converges to a perfect level with a sufficient number of samples, around 10^3 across kernels for this data. However, it is more complicated with the adversarial loss, varying greatly by kernel and number of samples, even when the attack size is fixed at one standard deviation of a given feature. Regardless of the number of samples, we see divergent time scales around 10^3 samples across all kernels.

6.3. Attack Time and Efficacy

Since PGD is iterative, we examined how increasing the raw compute time changes attack efficacy (Figure 4, Figure 3). It shows that there is no general relationship between attack time and induced error when we examine the entire attack space. The attacks that

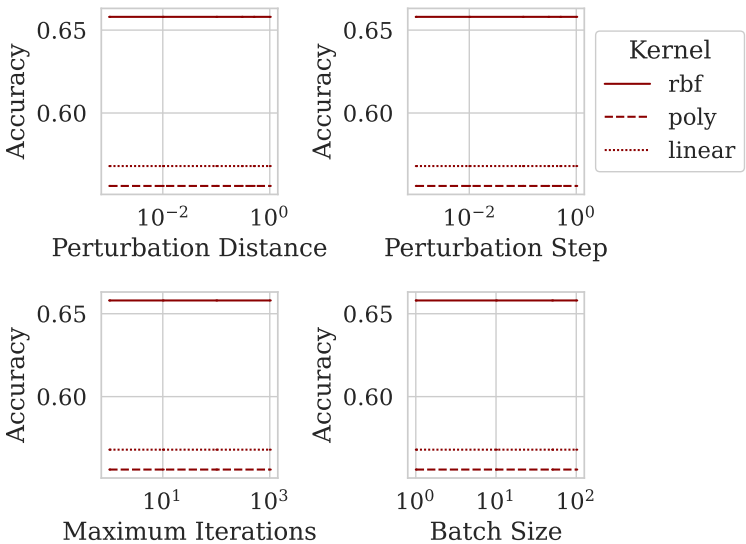


Figure 3. Attack Parameters vs Accuracy: This depicts how various attack hyperparameters change the accuracy.

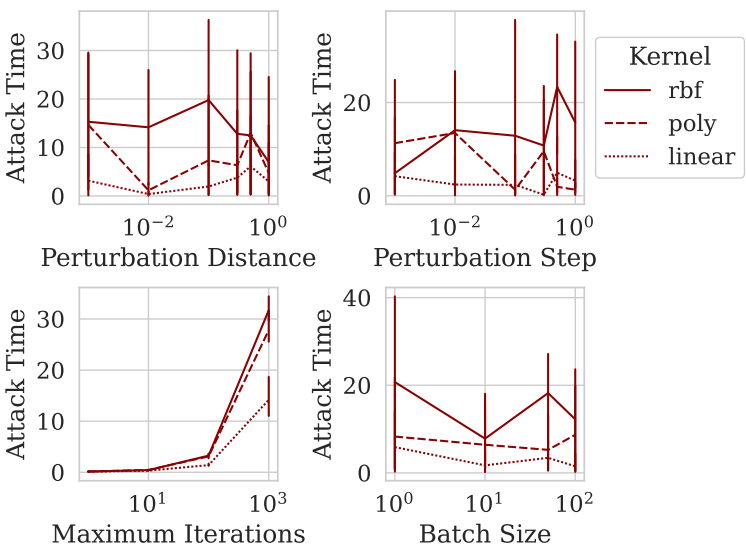


Figure 4. Attack Parameters vs Time: This depicts how various attack hyperparametrs change the run-time. The bars reflect the 95% confidence interval for all tested configurations.

produce the largest errors are more dependent on hyper-parameter choice than raw processing time (measured in iterations). This is highlighted further in (Figure 3), where we controlled for both step size and perturbation size, showing many examples where attack error was maximized with a small number of iterations. That is, a 'good' attack converges on strong adversarial examples quickly. Both plots illustrate that the polynomial kernel has a maximum error near 0.9, we see 0.95 for the RBF kernel, and perfect loss against the linear kernel. In general, we can verify that iterations have little to no effect (Figure 3) and that increasing batch size increases loss (Figure 3) to the detriment of false confidence (Figure 5c).

6.4. Critical Space:

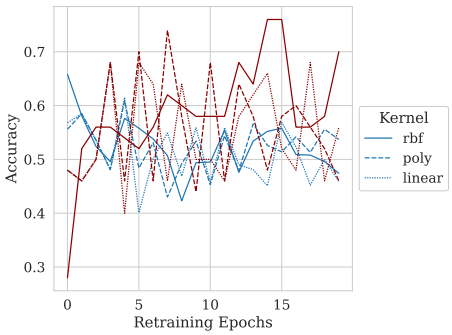
To reliably evaluate a model, we must look at how it performs across many attacks, so we measured both error (Figure 3) and false confidence (Figure 5c) for the entire attack space. We see that increasing either step size or perturbations tends to increase loss with a minimal perturbation size required by a given model and data-set. In addition, we found that there is a minimum perturbation value for effective attacks (around 0.1 for linear and polynomial kernels, and 0.5 for RBF), dependent on model and data (Figure 3). The effect of step size is much more variant, presumably dependent on the the other parameters. We also see that the RBF kernel is consistently the most robust against error (Figure 3), but this does little to stop false confidence (Figure 5c).

Attack time and error had a correlation of 0.18 suggesting that a clever choice of attack parameter is far more effective than adding raw processor time. This is further supported by the fact that step size has a correlation of 0.54 and perturbation distance has a correlation of 0.34. Even low-resolution, fast attacks can lead to maximized loss (Figure 3) raising further questions about the possibility of a truly reliable defense. The individual confidence for an example is inversely related to the size of the batch provided to the attacker, but only marginally (Figure 5c). Step size and total perturbation change confidence levels in complex ways beyond some critical point for either, but tend to converge. However, we can see that maximum confidence occurs when both step size and total perturbation are very small, creating a tension between highly confident attacks and attacks that produce maximum loss. This is intuitive—the loss is maximized by ensuring every sample crosses the decision boundary (even if only slightly) whereas confidence is maximized when perturbations put a sample in the center of the opposite class, usually far from the decision boundary. Adversarial retraining is an attempt defend against such perturbations.

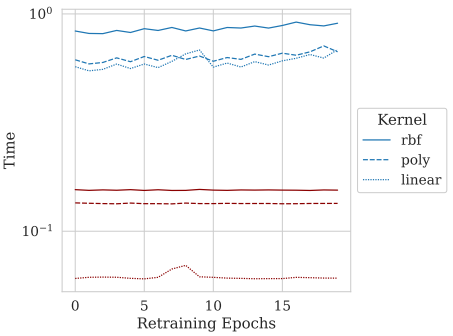
6.5. Adversarial Retraining

Adversarial Retraining. Adversarial retraining is a defense proposed by Li et al. [10], that appends adversarial examples to the training set, labels them 'malicious' and trains a classifier on a new set. This can be conducted iteratively, in 'epochs'. Figures 5a, and 5b depict this method conducted over 20 epochs on the RBF, polynomial, and linear kernels respectively. We can see that this process increases training time linearly, even when we exclude the attack generation time. The RBF and polynomial kernels do become more robust with successive epochs; however, this comes at the cost of benign accuracy. The linear kernel retains its benign accuracy with marginally improved robustness, but would still not reliably prevent false classifications. Unfortunately, as adversarial attacks blur the boundary between the 'benign' and 'malicious' sets, the number of support vectors tends to increase, leading to growth in time complexity beyond the time required by the larger number of samples while doing little to make reliable models. A related method, confidence calibrated retraining, attempts to solve this problem [3] but requires an additional iterative calculation that is guaranteed to increase run-time anyway. However, using the naive version, we found that the new classifier is susceptible to old attacks (compare the results Figure 5c and Figure 5d) despite reducing the efficacy of a given attack over many retraining cycles (see Figure 5a). In addition, we found that the models have nearly identical false confidence Figure 5d) on the attacks generated on the un-defended models (see Figure 5c), suggesting that the transferability of attacks has been under-estimated. This figure (Figure 5d) depicts all attacks that induce false classifications below the 99% true, benign detection threshold dictated by AT, even when this threshold is minimized. While the defended model (Figure 5d) has a better response against the strongest attack than the undefended models (Figure 3), this defense leads to a generalized failure on attacks (Figure 5d) relative to the benign model (Figure 3). Furthermore, strong attacks are possible across the entire attack space and work on nearly half of all examples. Even after the increased training time of AT, a strong attack (Figure 5d) was found in milliseconds.

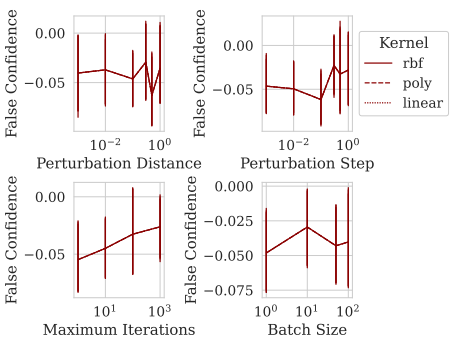
Furthermore, theoretical analysis shows that strong attacks will only cost more than model building for very large numbers of adversarial examples, which we've shown to be unnecessary when controlling for batch size. Since every model query has the potential to expose the attacker, a small number of queries is preferable anyway. Assuming the benign accuracy of 85% reflects the real-world behavior of the model, more 40% of queries will evade the classifier, suggesting that



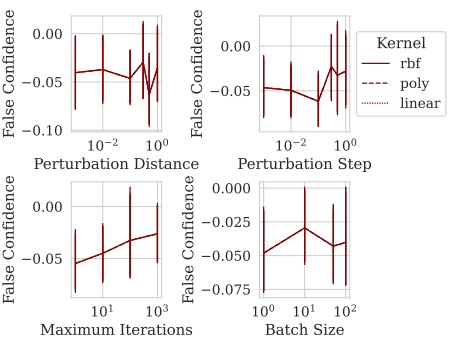
(a) Adversarial Retraining: Accuracy over several retraining cycles. The blue indicates the performance on unperturbed data and the red indicates performance on the adversarial data.



(b) Adversarial Retraining: Training and attack times (blue or red respectively) over several retraining cycles with $d_{max} = 1$



(c) False Negative Classifications Before Retraining



(d) False Negative Classifications after Retraining

Figure 5. This figure depicts the benign and adversarial accuracy (a), the training and attack times (b) and the false confidence across all attacks before (c) and after (d) adversarial retraining. The bars reflect the 95% confidence interval for all tested configurations.

an attacker armed with a large database of attacks will easily circumvent AT counter measures.

6.6. Limitations

Modern databases are measured in the millions and our evaluations fall below that by an order of magnitude. However, as we found that increasing the database size has an inconsistent effect on both benign or adversarial accuracy (Figure 2a) while substantially increasing run-time (Figure 2c). In order to conduct the wide number of experiments presented in a reasonable time, we used the smaller database size.

Section 3.1 demonstrates how this analysis extends to more complex models. Other machine learning methods could of course also be used (*e.g.* neural networks), but the main goal here was to examine the relative speed of attacks against polynomial time models more generally. Because our support vector machines require access to the entire data-set and create a set of support vectors that must be stored together in memory, we limited our tests to a single machine in order to minimize the complexities of network overhead.

While our analysis focuses on support vector machines, the increased run-time complexity of neural networks suggests that the cost-gap issue is even worse with modern models though other research [3, 54] has already noted this. While there is some remaining evidence for more effective model defences, for instance by using different forms of regularization [55, 56] or by modifying a neural network [57], both methods add run-time cost and do not necessarily offset the efficacy of an attacker, particularly if the step-size, batch-size, and number of iterations are well-tuned.

The parallel methodology could be extended to more sophisticated attacks without loss of generality.

Additionally, since 'good' attacks tend to work on most samples, further search optimizations that quickly eliminate bad attack candidates are possible. In addition, running the attack on multiple machines would reduce the load on the operating system relative to our single-machine scenario, but still favoring a relatively simple attacker over a large, complex, and centralized model generation process.

Despite any experimental limitations, both `optuna` and our code base support scaling across multiple machines and can easily be made to be 'massive' in a more traditional sense. However, it is clear that proper robustness evaluations require not only quantifying accuracy, but also require measuring feasible attack times and the confidence level of false classifications. In addition, models should be tested across the widest possible number of attack parameters since a given defense and data-set will change the efficacy of a given attack.

7. Conclusion

In this work, we propose a naive parallel implementation for evading classifiers in which the model gradients of SVMs and training data distributions are known to the attacker. While this level of information is hard to obtain in real-world scenarios, we highlight other research that proposes methods for obtaining this information from an otherwise obscured model or dataset. We demonstrated that a well-chosen step size will add more strength to an attack than raw processing time. We confirmed earlier observations that accuracy and robustness are inversely related. We also show that model-building is computationally more expensive than attacks, especially in the context of adversarial retraining. Despite the optimistic results in published work, we find that perturbing a sample's features by only a single standard deviation is sufficient to reliably break classifiers while adversarial retraining as dictated by NIST standards[24]. While this degree of perturbation may create obvious adversaries to humans, our best attempts to automatically detect them still resulted in decreased benign accuracy, higher training times, and a failure to prevent false classifications. Finally, we provide an easily extensible code-base for managing massive, parallel, and distributed experiments on various attacks and defences. Thus, we find adversarial retraining to be unsuitable for real-time, safety-critical, or security-sensitive applications of KSVMs. Simultaneously, through a run-time analysis of low-cost model (KSVMs), we raise serious concerns about the hope of any polynomial-time model builder to defend against an adversary that consistently succeeds in more-or-less constant time, despite many rounds of adversarial retraining. Furthermore, we show this to be true on generated data, system process data [49], and social media data [51] (see: Section 6 and Appendices A & B for each dataset, respectively) .

8. Acknowledgements

Financial support has been provided in part by the Knut and Alice Wallenberg Foundation grant number 2019.0352 and by the eSSENCE Programme under the Swedish Government's Research Initiative.

References

[1] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. and Fergus, R. (2013) Intriguing properties of neural networks. *International Conference on Learning Representations* .

[2] Madry, A., Makelov, A., Schmidt, L., Tsipras, D. and Vladu, A. (2017) Towards deep learning models resistant to adversarial attacks. *International Conference on Machine Learning* .

[3] Croce, F. and Hein, M. (2020) Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *International Conference on Machine Learning* .

[4] Deka, P.K., Bhuyan, M.H., Kadobayashi, Y. and Elmroth, E. (2019) Adversarial impact on anomaly detection in cloud datacenters. In *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)* (IEEE): 188–18809.

[5] Kim, D.S. and Park, J.S. (2003) Network-based intrusion detection with support vector machines. In *International Conference on Information Networking* (Springer): 747–756.

[6] Mehmood, T. and Rais, H.B.M. (2015) Svm for network anomaly detection using aco feature subset. In *2015 International symposium on mathematical sciences and computing research (iSMSC)* (IEEE): 121–126.

[7] Tzotsos, A. and Argialas, D. (2008) Support vector machine classification for object-based image analysis. In *Object-Based Image Analysis* (Springer), 663–677.

[8] Uesato, J., O'Donoghue, B., Oord, A.v.d. and Kohli, P. (2018) Adversarial risk and the dangers of evaluating against weak attacks. *Proceedings of Machine Learning Research* .

[9] Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I. *et al.* (2019) On evaluating adversarial robustness. *arXiv:1902.06705* .

[10] Li, B., Vorobeychik, Y. and Chen, X. (2016) A general retraining framework for scalable adversarial classification. *Workshop on Adversarial Training, Neural Information Processing Systems* .

[11] Stutz, D., Hein, M. and Schiele, B. (2019) Confidence-calibrated adversarial training: Towards robust models generalizing beyond the attack used during training. *International Conference on Machine Learning* .

[12] Demontis, A., Melis, M., Pintor, M., Jagielski, M., Biggio, B., Oprea, A., Nita-Rotaru, C. *et al.* (2019) Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *28th {USENIX} Security Symposium*: 321–338.

[13] Desislavov, R., Martínez-Plumed, F. and Hernández-Orallo, J. (2021) Compute and energy consumption trends in deep learning inference. *arXiv:2109.05472* .

[14] Tsipras, D., Santurkar, S., Engstrom, L., Turner, A. and Madry, A. (2018) Robustness may be at odds with accuracy. *Int'l Conference on Learning Representations* .

[15] Raghunathan, A., Xie, S.M., Yang, F., Duchi, J. and Liang, P. (2020) Understanding and mitigating the tradeoff between robustness and accuracy. *International Conference on Machine Learning* .

[16] Dohmatob, E. (2019) Generalized no free lunch theorem for adversarial robustness. In *International Conference on Machine Learning* (PMLR): 1646–1654.

[17] Carlini, N. and Wagner, D. (2017) Towards evaluating the robustness of neural networks. In *IEEE symposium on security and privacy (sp)* (IEEE): 39–57.

[18] Athalye, A., Carlini, N. and Wagner, D. (2018) Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *Int. Conference on Machine Learning* .

[19] Cortes, C. and Vapnik, V. (1995) Support-vector networks. *Machine learning* **20**(3): 273–297.

[20] Trafalis, T.B. and Gilbert, R.C. (2007) Robust support vector machines for classification and computational issues. *Optimisation Methods and Software* **22**(1): 187–198.

[21] Bordes, A., Ertekin, S., Weston, J. and Bottou, L. (2005) Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research* **6**(Sep): 1579–1619.

[22] Christmann, A. and Steinwart, I. (2004) On robustness properties of convex risk minimization methods for pattern recognition. *The Journal of Machine Learning Research* **5**: 1007–1034.

[23] Bienstock, D., Muñoz, G. and Pokutta, S. (2018) Principled deep neural network training through linear programming. *arXiv:1810.03218* .

[24] Falco, J.A., Hurd, S. and Teumim, D. (2006) *Using host-based anti-virus software on industrial control systems: Integration guidance and a test methodology for assessing performance impacts* (NIST).

[25] Organization, I.S. (2018), ISO 26262-1:2011, road vehicles — functional safety, https://www.iso.org/standard/43464.html (visited 2022-04-20).

[26] Su, J., Vargas, D.V. and Sakurai, K. (2019) One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* **23**(5): 828–841.

[27] Chen, J., Jordan, M.I. and Wainwright, M.J. (2020) Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)* (IEEE): 1277–1294.

[28] Brown, T.B., Mané, D., Roy, A., Abadi, M. and Gilmer, J. (2017) Adversarial patch. *arXiv:1712.09665* .

[29] Brendel, W., Rauber, J. and Bethge, M. (2017) Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv:1712.04248* .

[30] Liu, X., Yang, H., Liu, Z., Song, L., Li, H. and Chen, Y. (2018) Dpatch: An adversarial patch attack on object detectors. *arXiv:1806.02299* .

[31] Qin, Y., Carlini, N., Cottrell, G., Goodfellow, I. and Raffel, C. (2019) Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In *International conference on machine learning* (PMLR): 5231–5240.

[32] Grosse, K., Pfaff, D., Smith, M.T. and Backes, M. (2018) The limitations of model uncertainty in adversarial settings. *arXiv:1812.02606* .

[33] Kotyan, S. and Vargas, D.V. (2019) Adversarial robustness assessment: Why both l_0 and l_∞ attacks are necessary. *arXiv:1906.06026* .

[34] Chen, P.Y., Zhang, H., Sharma, Y., Yi, J. and Hsieh, C.J. (2017) Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*: 15–26.

[35] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. and Talwalkar, A. (2017) Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research* **18**(1): 6765–6816.

[36] Hansen, N. (2016) The cma evolution strategy: A tutorial. *arXiv:1604.00772* .

[37] Ozaki, Y., Tanigaki, Y., Watanabe, S., Nomura, M. and Onishi, M. (2022) Multiobjective tree-structured parzen estimator. *Journal of Artificial Intelligence Research* **73**: 1209–1250.

[38] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M. *et al.* (2011) Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12**: 2825–2830.

[39] Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G. *et al.* (2013) Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases* (Springer): 387–402.

[40] Liu, T., Wang, H., Li, L., Luo, X., Dong, F., Guo, Y., Wang, L. *et al.* (2020) MadDroid: Characterizing and detecting devious ad contents for android apps. *Proceedings of The Web Conference 2020* .

[41] Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D. and Elovici, Y. (2018) N-BaIoT—network-based detection of IoT botnet attacks using deep autoencoders. *IEEE Pervasive Computing* **17**(3): 12–22.

[42] Wang, X., Li, J., Kuang, X., Tan, Y.a. and Li, J. (2019) The security of machine learning in an adversarial setting: A survey. *Journal of Parallel and Distributed Computing* **130**: 12–23.

[43] Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A. and Mukhopadhyay, D. (2018) Adversarial attacks and defences: A survey. *arXiv:1810.00069* .

[44] Biggio, B., Nelson, B. and Laskov, P. (2012) Poisoning attacks against support vector machines. *International Conference on Machine Learning* .

[45] Tramèr, F., Zhang, F., Juels, A., Reiter, M.K. and Ristenpart, T. (2016) Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium Security 16)*: 601–618.

[46] Fredrikson, M., Jha, S. and Ristenpart, T. (2015) Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd*

ACM SIGSAC Conference on Computer and Communications Security: 1322–1333.

[47] Ateniese, G., Mancini, L.V., Spognardi, A., Villani, A., Vitali, D. and Felici, G. (2015) Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks* **10**(3): 137–150.

[48] Shokri, R., Stronati, M., Song, C. and Shmatikov, V. (2017) Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)* (IEEE): 3–18.

[49] Tavallaee, M., Bagheri, E., Lu, W. and Ghorbani, A.A. (2009) A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications* (Ieee): 1–6.

[50] Dua, D. and Graff, C. (2017), UCI machine learning repository. URL `http://archive.ics.uci.edu/ml`.

[51] Dadkhah, S., Zhang, X., Weismann, A.G., Firouzi, A. and Ghorbani, A.A. (2023) TruthSeeker: The Largest Social Media Ground-Truth Dataset for Real/Fake Content doi:10.36227/techrxiv.22795130.v1, URL `https://www.techrxiv.org/articles/preprint/TruthSeeker_The_Largest_Social_Media_Ground-Truth_Dataset_for_Real_Fake_Content/22795130`.

[52] Akiba, T., Sano, S., Yanase, T., Ohta, T. and Koyama, M. (2019) Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*: 2623–2631.

[53] Chang, C.C. and Lin, C.J. (2011) Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)* **2**(3): 1–27.

[54] Meyers, C., Löfstedt, T. and Elmroth, E. (2023) Safety-critical computer vision: an empirical survey of adversarial evasion attacks and defenses on computer vision systems. *Artificial Intelligence Review* : 1–35.

[55] Jakubovitz, D. and Giryes, R. (2018) Improving dnn robustness to adversarial attacks using jacobian regularization. In *Proceedings of the European Conference on Computer Vision (ECCV)*: 514–529.

[56] Ross, A. and Doshi-Velez, F. (2018) Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, **32**.

[57] Colbrook, M.J., Antun, V. and Hansen, A.C. (2021) Can stable and accurate neural networks be computed. *On the barriers of deep learning and Smale's 18th problem. arXiv* **2101**.

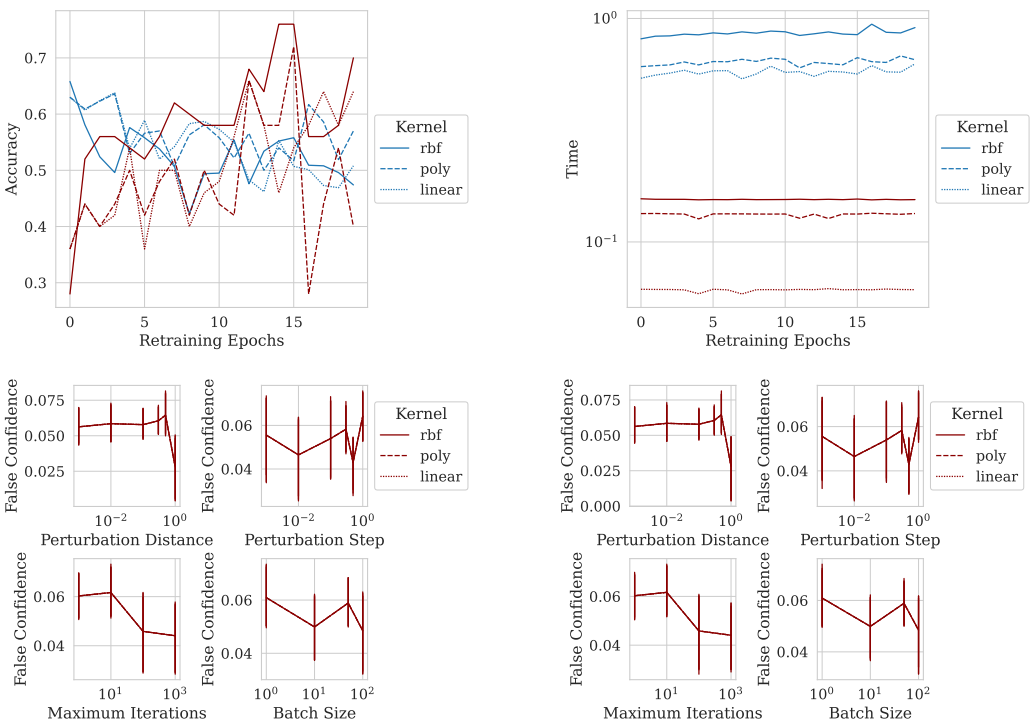


Figure A.1. Efficacy of Adversarial Retraining on KDD-NSL Dataset. The top left depicts the adversarial and benign accuracy over a number of retraining epochs. The top right depicts the per epoch training time as the number of training epochs increases. The bottom row depicts the false confidence before retraining (left) on strong adversarial examples and after (right). The bars reflect the 95% confidence interval for all tested configurations.

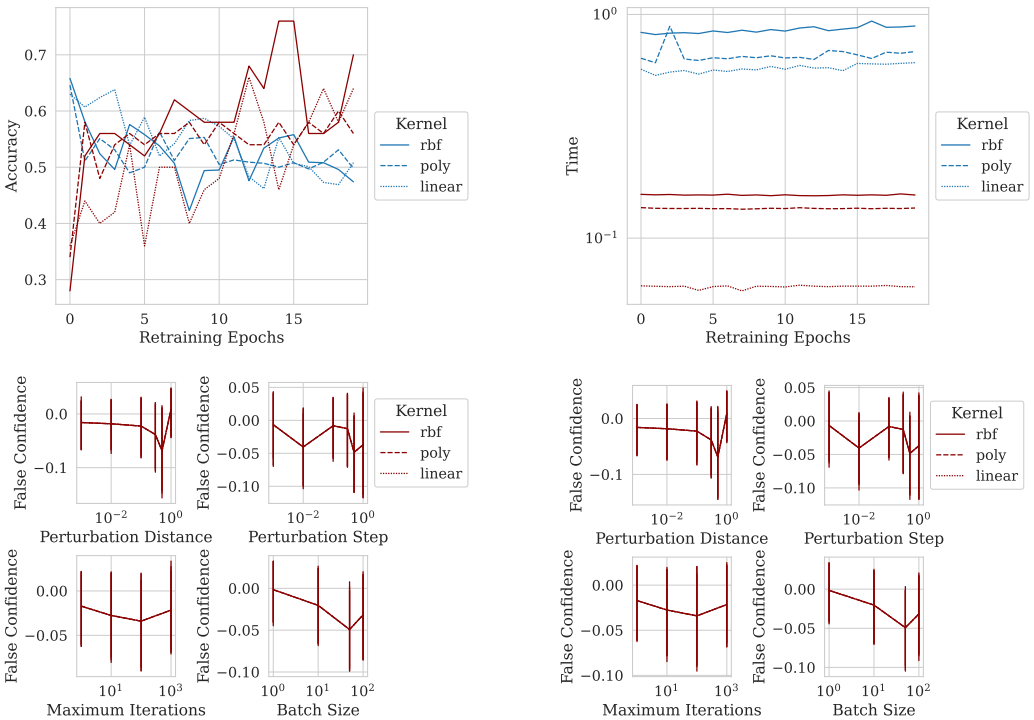


Figure B.2. Efficacy of Adversarial Retraining on Truthseeker Dataset. The top left depicts the adversarial and benign accuracy over a number of retraining epochs. The top right depicts the per epoch training time as the number of training epochs increases. The bottom row depicts the false confidence before retraining (left) on strong adversarial examples and after (right). The bars reflect the 95% confidence interval for all tested configurations.

Paper III

Training Rate and Survival Heuristic for Inference and Robustness Evaluation (Trashfire).

Charles Meyers, Mohammad Reza Saleh Sedghpour, Tommy Löfstedt, and Erik Elmroth.

IEEE International Conference on Machine Learning and Cybernetics (ICMLC): Volume 1, 2024.

How can we formalise the cost benefit relationship between robustness and particular choice of model?

A TRAINING RATE AND SURVIVAL HEURISTIC FOR INFERENCE AND ROBUSTNESS EVALUATION (TRASHFIRE)

CHARLES MEYERS[1], MOHAMMAD REZA SALEH SEDGHPOUR[2,1], TOMMY LÖFSTEDT[1], ERIK ELMROTH[1,2]

[1]Department of Computing Science, Umeå University, Umeå, Sweden
[2]Elastisys AB
E-MAIL: cmeyers@cs.umu.se, msaleh@cs.umu.se, tommy@cs.umu.se, elmroth@cs.umu.se

Abstract:

Machine learning models—deep neural networks in particular—have performed remarkably well on benchmark datasets across a wide variety of domains. However, the ease of finding adversarial counter-examples remains a persistent problem when training times are measured in hours or days and the time needed to find a successful adversarial counter-example is measured in seconds. Much work has gone into generating and defending against these adversarial counter-examples, however the relative costs of attacks and defences are rarely discussed. Additionally, machine learning research is almost entirely guided by test/train metrics, but these would require billions of samples to meet industry standards. The present work addresses the problem of understanding and predicting how particular model hyper-parameters influence the performance of a model in the presence of an adversary. The proposed approach uses survival models, worst-case examples, and a cost-aware analysis to precisely and accurately reject a particular model change during routine model training procedures rather than relying on real-world deployment, expensive formal verification methods, or accurate simulations of very complicated systems (*e.g.*, digitally recreating every part of a car or a plane). Through an evaluation of many pre-processing techniques, adversarial counter-examples, and neural network configurations, the conclusion is that deeper models do offer marginal gains in survival times compared to more shallow counterparts. However, we show that those gains are driven more by the model inference time than inherent robustness properties. Using the proposed methodology, we show that ResNet is hopelessly insecure against even the simplest of white box attacks.

Keywords:

Machine Learning; Computer Vision; Neural Networks; Adversarial AI; Trustworthy AI

1 Introduction

Machine Learning (ML) has become widely popular for solving complex prediction problems across many disciplines, such as medical imaging [20], computer security [2], aviation [36], and security [41]. Despite this, adversarial attacks exploit ML models by introducing subtle modifications to data which leads to misclassification or otherwise erroneous outputs [11]. To ensure the robustness of ML models against adversaries has become a critical concern [8, 10, 12, 44, 42].

The purpose of this work was to evaluate if survival analysis can predict the success of a particular set of model hyper-parameters. In addition, we explored the relationship between computational cost and prediction accuracy in both benign and adversarial contexts. By using samples crafted specifically to be challenging and applying survival models (see Section 3) we provide a framework to predict the expected failure time across the adversarial space. Using survival models, we demonstrate that larger machine learning models, while offering marginal gains over smaller models, do so at the expense of training times that far outpace the expected survival time and that it is simply not feasible to defend against certain attacks using the examined models and defences.

1.1 Motivations

It is routine to consider an adversarial context in safety—or security—critical applications [20, 2, 41] where we assume the attacker is operating in their own best-case scenario [32, 25, 35, 28, 43]. Cryptography often defines 'broken' in the context of time to quantify the feasibility of an attack [32]—'broken' algorithms are usually defined as those for which attacks can be conducted in a (relatively) small amount of time. For example, one recent study [25] distilled the process of password-cracking into a cloud-based service that can break common password schemes in a number of days. However, someone attacking a machine learning model might have a variety of competing goals (*e.g.*, minimising the perturbation distance or maximising the false confidence) [35, 12, 28, 22, 43], so time analyses are less straightforward. What is missing, however, is a method to directly model the effect of attack criteria on the survival time.

Much work has gone into mitigating adversarial attacks, for example by adding noise in the training process [55, 9], rejecting low-confidence results [13], or by reducing the bit-depth of the data and model weights [53]. However, these analyses focus on ad-hoc posterior evaluations on benchmark datasets (*e.g.*, CIFAR-10 or MNIST) to determine whether or not a given technique is more or less effective than another. That is, the relationship between marginal benefit and marginal cost is unclear. Furthermore, the community has trended towards larger models [18] and larger datasets [18, 4]. For example, autonomous vehicles still largely rely on system integration tests to verify safety [24], assuming that human-like accident metrics will guarantee safety. While there are simulation techniques [21] that highlight problematic scenarios by testing a component in a simulated world in which all components are modelled digitally, implementing them requires building an entire digital world that can nevertheless miss real-world edge cases. Furthermore, while formal methods for neural network verification do exist, they are generally too costly to be feasible for tuning and verifying large scale machine learning models [40]. To reach safety-critical standards that are routine in other industries [39, 37, 38], the machine learning field must move beyond the limited test/train split paradigm that would require many, many billions of test samples for every change of a model to meet industry standards [42]. The proposed method models the complex relationship between model hyperparameters and the resulting robustness of the model, using nothing more than routine metrics collected in the model tuning stage.

1.2 Contributions

The contributions of this work are:

- Survival analysis models for analysing ML models under adversarial perturbations with substantial empirical evidence that survival analysis is both effective and dataset-agnostic, allowing for the expected failure rate to be predicted more precisely and accurately.
- Survival analysis models to measure model robustness across a wide variety of signal pre-processing techniques, exploring the relationships between latency, accuracy, and model depth.
- A novel metric: The *training rate and survival heuristic* (TRASH) *for inference and robustness evaluation* (FIRE) to evaluate whether or not a model is robust to adversarial attacks in a time- and compute-constrained context.
- Substantial empirical evidence that larger neural networks increase training and prediction time while adding little-to-no benefit in the presence of an adversary.

2 Background

Much work has gone into explaining the dangers of adversarial attacks on ML pipelines [10, 28, 22, 6], though studies on adversarial robustness have generally been limited to ad-hoc and posterior evaluations against limited sets of attack and defence parameters, leading to results that are, at best, optimistic [42]. Previous work on neural network verification have relied on expensive integration tests [24], elaborate simulation environments [21], or methods that are too computationally expensive to be useful for model selection [40]. However, the present work formalises methods to model the effect of attacks and defences on a given ML model and reveals a simple cost-to-performance metric to quickly discard ineffective strategies.

2.1 Adversarial Attacks

In the context of ML, an adversarial attack refers to deliberate and malicious attempts to manipulate the behaviour of a model. The presented work focused on *evasion attacks* that attempt to induce misclassifications at run-time [10, 6], but note that the proposed methodology (Section 3) and cost analysis (Section 4) extends to other types of attacks, such as database poisoning [5, 46], model inversion [14, 33], data stealing [45], or denial of service [47]. In all sections below, metrics were

collected on the benign (unperturbed) data and adversarial (perturbed) data. The abbreviations *ben* and *adv* are used throughout, respectively. The strength of an attack is often measured in terms of a perturbation distance [11, 28]. The perturbation distance, denoted by $\varepsilon \geq 0$, quantifies the magnitude of the perturbation applied to a sample, x, when generating a new adversarial sample, x'. The definition is,

$$\varepsilon := \|x' - x\| \leq \varepsilon^*, \tag{1}$$

where $\|\cdot\|$ denotes a norm or pseudo-norm (*e.g.*, the Euclidean ℓ_2 norm or the ℓ_0 pseudo-norm). We denote by ε^* the maximum allowed perturbation of the original input. For example, this might be one bit, one pixel, or one byte, depending on the test conditions. For more information on different criteria, see Section 5.4.

2.1.1 Accuracy and Failure Rate

The accuracy refers to the percentage or proportion of examples that are correctly classified. A lower accuracy indicates a higher rate of misclassifications or incorrect predictions. The accuracy, Acc, is defined as

$$\text{Acc} := 1 - \frac{\text{False Classifications}}{N}, \tag{2}$$

where N is the total number of samples. The accuracy on a given test set, presumed to be drawn from the same distribution as the training set, is called the *benign accuracy*, Acc_{ben}. The *adversarial accuracy*, Acc_{adv}, is a measure of correct classifications in the presence of noise intended to be adversarial. However, accuracy is known to vary according to the model hyper-parameters [49] and various run-time considerations. Therefore, it is useful to think in terms of *failure rate*, as

$$\text{Failure Rate} := \frac{\text{False Classifications}}{\Delta t}, \tag{3}$$

where Δt is a time interval. By parameterizing the measure of misclassification by time, it is possible to model the chance of failure as a function of various attributes and parameters of a model.

Let h be a function that describes the rate of failure at time t. This is a way to express the failure rate in terms of a *hazard function*, which is defined as

$$h(t) := \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t \,|\, T \geq t)}{\Delta t}, \tag{4}$$

where P is a probability and T is the time until a false classification occurs, also referred to as *survival time* [26]. To be able to compare the computational efficacy of different model and attack configurations, we modelled the probability of not observing a failure before a given time, t, using the *cumulative hazard function*,

$$H(t) := \int_0^t h(\tau)\, d\tau. \tag{5}$$

Then, the *cumulative survival function* is

$$S(t) := P(T \geq t) = \exp(-H(t)) = 1 - F(t) = 1 - \int_0^t f(u) du \tag{6}$$

where $F(t)$ is the *lifetime distribution function* which describes the cumulative probability of failure before time t, or $F(t) = P(T < t)$. The probability density of observing a failure at time, t, is [26, 15],

$$f(t) := h(t)S(t).$$

In practice, the $h(t), S(t)$, and/or $f(t)$ can be determined whenever one of them is known [26].

Survival analysis models have been widely used to investigate the likelihood of failures across fields where safety is a primary concern (*e.g.*, in medicine, aviation, or auto-mobiles) [34, 31]. These models allow us to examine the effect of the specified covariates on the failure rate of the classifier. For manufacturing, this is done by simulating normal wear and tear on a particular hardware component (*e.g.*, a motor or aircraft sensor) [34] by exposing the component to vibration, temperatures, or impacts. For the study of diseases in humans, these models are often build on demographic data and used to examine the effect of things like age, gender, and/or treatment on the expected survival time of a patient. Likewise, survival analysis can be used to estimate the time until a successful adversarial attack of an ML pipeline or component using metrics that are routinely collected as part of normal model training procedures. The covariates, for example, might be things like perturbation distance, model depth, number of training epochs, a signal processing technique, *etc.*

2.2 Cost

Assume that the cost of training a model, C_{train}, is a function of the total training time, T_{train}, the number of training samples, N_{train}, and the training time per sample, $t_{\text{train}} = \frac{T_{\text{train}}}{N_{\text{train}}}$, such that the cost of training on hardware with a fixed time-cost is

$$C_{\text{train}} := C_h \cdot T_{\text{train}} = C_h \cdot t_{\text{train}} \cdot N_{\text{train}}, \tag{7}$$

where C_h is the cost per time unit of a particular piece of hardware. Hence, the cost is assumed to scale linearly with per-sample training time and sample size, N_{train}. Analogously, t_{predict} is used elsewhere in this text to refer to the prediction time for a set of samples, divided by the number of samples. Assuming the attacker and model builder are using similar hardware, then the cost to an attacker, C_{attack}, is $C_{\text{attack}} := C_h \cdot T_{\text{attack}} = C_h \cdot t_{\text{attack}} \cdot N_{\text{attack}}$, where N_{attack} is the number of attacked samples. Furthermore, a fast attack will be lower-bounded by the model inference time, t_{predict}, which is generally much smaller than the training time, t_{train}. Of course, the long-term costs of deploying a model will be related to the inference cost, but a model is clearly broken if the cost of improving a model ($\propto t_{\text{train}}$) is larger than the cost of finding a counterexample ($\propto t_{\text{attack}}$) within the bounds outlined in Equation 1. The training cost per sample does not consider how well the model performs, and a good model is one that both generalises and is reasonably cheap to train. Therefore, a cost-normalised failure rate metric is introduced in Equation 9 in Section 4. Before comparing this cost to the failure rate, the attack time per sample—or the expected survival time—must be estimated. For that, survival models can be used.

3 Survival Analysis for ML

Failure time analysis has been widely explored in other fields [7], from medicine to industrial quality control [20, 27, 41], but there is very little published research in the context of ML. However, as noted by many researchers [35, 10, 42], these models are fragile to attackers that intend to subvert the model, steal the database, or evade detection. In this work, we leverage evasion attacks to examine the parameterised time-to-failure—or survival time—denoted $S_\theta(t)$, where θ is a set of parameters that describe the joint effect of the covariates on the survival time, usually found through maximum likelihood estimation on observed survival data [15]. All survival models can be expressed in terms of this parameterised survival function, $S_\theta(t)$, hazard function, $H_\theta(t)$, and lifetime probability distribution, $F_\theta(t)$, such that

$$S_\theta(t) := \exp\big\{ -H_\theta(t)\big\} := 1 - F_\theta(t) := 1 - \int_0^t f_\theta(u)du,$$

and the expected survival time is thus

$$\mathbb{E}_{S_\theta}[T] = \int_0^{t^*} S_\theta(u)du \approx t_{\text{attack}},$$

where t_{attack} is an estimate of the time it takes for the average attacker to induce a failure subject to the condition in Equation 1 and t^* is the latest observed time (regardless of failure or success). The parameters, θ, are estimated from model evaluation data such that: $h_\theta(t = t_{attack}) \approx 1 - \text{Acc}_{\text{adv}}$.

Survival analysis models have been widely used to investigate the likelihood of failures across fields where safety is a primary concern (*e.g.*, in medicine, aviation, or auto-mobiles) [34, 31]. These models allow us to examine the effect of the specified covariates on the failure rate of the classifier. These survival analysis models can broadly be separated into two categories: proportional hazard models and accelerated failure time models, each of which is outlined in the subsections below. Furthermore, by parameterizing the performance by time, it is possible to do a cost-value analysis, as outlined in Section 4.

3.1 The Cox Proportional Hazard Model

The Cox proportional hazard model tries to find model parameters, θ, corresponding to covariates, x, to predict the hazard function on unseen configurations of the covariates, such that

$$h_\theta(t) = h_0(t)\phi_\theta(x) = h_0(t)\exp(\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p),$$

where θ_i is the i-th model parameter and x_i is the measurement of the i-th covariate. One downside of the Cox model compared to the accelerated failure time models discussed below is that there are few distributions that fit the proportional hazards assumption. A common choice is therefore to use a non-parametric approximation of the baseline hazards function [15]. Additionally, unlike the accelerated failure time models discussed below, the coefficients in the Cox model, θ, are inter-dependent (they are said to be *adjusted* for each other) and, as such, their interpretation is not straightforward [15].

3.2 Accelerated Failure Time Models

While Cox models assume that there is a multiplicative effect on the baseline hazard function, h_0, due to the effect of a covariate, accelerated failure time (AFT) models instead assume that the effect of a covariate is to accelerate or decelerate the time in a baseline survival function, $S_0(t)$. Accelerated failure time models have the form

$$S_\theta(t) = S_0\left(\frac{t}{\phi_\theta(x)}\right). \tag{8}$$

Unlike the proportional hazard model discussed above, the coefficients of AFT models have a straightforward interpretation where a value of θ represents an θ-fold increase in failure risk [15] and a negative value indicates a corresponding decrease in failure risk. The survival function is commonly derived using *e.g.*, the exponential, Weibull, log-normal,

log-logistic, or generalised gamma distributions [15]—each of which was tested in this work.

3.3 Survival Model Validation

To compare the efficacy of different parametric AFT models, we use the Akaike information criterion (AIC) and the Bayesian information criterion (BIC) [51], where the preferred model will be the one with the smallest value. We provide the concordance score, which gives a value between 0 and 1 that quantifies the degree to which the survival time is explained by the model, where a 1 reflects a perfect explanation [52] and 0.5 reflects random chance. We also include two measures of error between the fitted model (predictions) and a model fit to the data using a cubic spline (observations) as proposed by Austin *et al.* [3]. The first such measure of error is the mean difference between the predicted and observed failure probabilities, called the integrated calibration index (ICI). The second metric is the error between these curves at the 50$^{\text{th}}$ percentile (E50) [3]. Except for AIC and BIC, we have provided these metrics for both the training and test sets, the latter of which was 20% of the total number of samples.

4 Failure Rates and Cost Normalisation

With an estimate for the expected survival time, the cost-normalised failure rate, or training time to attack time ratio, can be quantified. Under the assumption that the cost scales linearly with t_{train} (as in Equation 7), one can divide this cost by the expected survival time to get a rough estimate of the relative costs for the model builder ($C_{\text{train}} \propto t_{\text{train}}$) and the attacker ($C_{\text{adv.}} \propto t_{\text{attack}} \approx \mathbb{E}_{S_\theta}[T]$). Recalling the definition of ε in Equation 1, the cost of failure in adversarial terms can be expressed as,

$$\bar{C}_{\text{adv.}} = \frac{t_{\text{train}}}{\mathbb{E}_\theta[T \mid 0 < \varepsilon \leq \varepsilon^*]}. \tag{9}$$

If $\bar{C}_{\text{adv.}} \gg 1$ then the model is *broken* since it is cheaper to attack the model than it is to train it. The numerator can be thought as the approximate training time per sample, or training rate, and the denominator is the expected survival heuristic. The ratio of these allows one to quantify the comparative cost of the model builder and the attacker and the coefficients of the survival model provide a way to estimate the effects of the covariates. We call this metric the TRASH score since it quickly indicates whether more training is likely to improve the adversarial robustness and any score > 1 indicates that a given model is, in fact, irredeemable.

5 Methodology

Below we outline the experiments performed and the hyper-parameter configurations of the models, attacks, and defences across the various model architectures, model defences, and attacks. All experiments were conducted on Ubuntu 18.04 in a virtual machine running in a shared-host environment with one NVIDIA V100 GPU using Python 3.8.8. All configurations were tested in a grid search using `hydra` [54] to manage the parameters, `dvc` [19] to ensure reproducibility, and `optuna` [1] to manage the scheduling. For each attack and model configuration, the metrics outlined in Equations 2–9 were collected, as well as the inference time, training time, and attack generation time. A grid search was conducted over datasets, models, defences, and attacks across ten permutations of the data. For visualisation, the $f_{\text{ben.}}$ and $f_{\text{adv.}}$ were approximated for each attack and defence combination using Equation 3, and $\bar{C}$ was approximated in the adversarial case as per Equation 9. Additionally, we provide links to the source code repository[1], as well as the source for this document and archived data[2]

5.1 Dataset

Experiments were performed on both the CIFAR100, CIFAR10 [29], and MNIST [17] datasets. The adversarial and benign accuracies were measured together with the attack generation time and the prediction time. Equations 3 and 9 were used to calculate the adversarial failure rate and the cost. For accuracy, see Equation 2. For training, 80% of the samples were used for all datasets. Of the remaining 20%, one-hundred class-balanced samples were selected to evaluate each attack. In addition, all data were shuffled to provide ten training and test sets for each hyper-parameter combination. Then, the data were centred and scaled (using statistics computed from the training set to avoid data leakage). This provides a straight forward interpretation of ε, where $\varepsilon = 1$ implies one standard-deviation of noise.

5.2 Tested Models

The Residual Neural Network (ResNet) [23] is a popular classification model[3] because of its ability to train neural networks with many layers efficiently by using *residual connections*. The residual connections allow models to have hundreds of layers rather than tens of layers [23, 50]. Despite

[1] Our Source Code

[2] LaTeX source and data for this document.

[3] More than 180 thousand citations: ResNet citations on Google Scholar.

the prevalence of the reference architecture, several modifications have been proposed that trade off, for instance, robustness and computational cost by varying the number of convolutional layers in the model. We tested the *ResNet-18*, *-34*, *-51*, *-101*, and *-152* reference architectures, that get their names from their respective number of layers. We used the the `pytorch` framework and the Stochastic Gradient Descent minimiser with a momentum parameter of 0.9 and learning rates $\in \{10, 1, 0.1, 0.01, 0.001, 0.0001, .00001, 0.000001\}$ for epochs $\in \{10, 20, 30, 50, 100\}$.

5.3 Tested Defences

In order to simulate various conditions affecting the model's efficacy, we have also tested several defences that modify the model's inputs or predictions in an attempt to reduce its susceptibility to adversarial perturbations. Just like with the attacks, we used the Adversarial Robustness Toolbox [44] for their convenient implementations. The evaluated defences follow.

Gauss-in (ℓ_2): The 'Gaussian Augmentation' defence adds Gaussian noise to some proportion of the training samples. Here, we set this proportion to 50%, allowing to simulate the effect of noise on the resulting model [55]. Noise levels in $\{.001, .01, .1, .3, .5, 1\}$ were tested.

Conf (ℓ_∞): The 'High Confidence Thresholding' defence only returns a classification when the specified confidence threshold is reached, resulting in a failed query if a classification is less certain. This allows to simulate the effects of rejecting 'adversarial' or otherwise 'confusing' queries [13] that fall outside the given confidence range by ignoring ambiguous results without penalty. Confidence levels in $\{.1, .5, .9, .99, .999\}$ were tested.

Gauss-out (ℓ_2): The 'Gaussian Noise' defence, rather than adding noise to the input data, adds noise during inference [9], allowing to reduce precision to grey- and black-box attacks without going through costly training iterations. Noise levels in $\{.001, .01, .1, .3, .5, 1\}$ were tested.

FSQ: The 'Feature Squeezing' defence changes the bit-depth of the input data to minimise the noise induced by floating-point operations. It was included here to simulate the effects of various GPU or CPU architectures, which may also vary in bit-depth [53]. Bit-depths in $\{2, 4, 8, 16, 32, 64\}$ were tested.

5.4 Tested Attacks

Several attacks using the Adversarial Robustness Toolbox [44] were evaluated in order to simulate attacks that vary in information and run-time requirements across distance metrics. Other researchers have noted the importance of testing against multiple types of attacks [10]. For the purposes here, *attack strength* refers to the degree to which an input is modified by an attacker, as described in Section 1. Below is a brief description of the attacks that were evaluated. One or more norms or pseudo-norms were used in each attack, as given in the parentheses next to the attack name.

FGM ($\ell_1, \ell_2, \ell_\infty$): The 'Fast Gradient Method' quickly generates a noisy sample, with no feasibility conditions beyond a specified step size and number of iterations [22]. It generates adversarial samples by using the model gradient and taking a step of length ε in the direction that maximises the loss with $\varepsilon \in \{.001, .01, .03, .1, .2, .3, .5, .8, 1\}$.

PGD ($\ell_1, \ell_2, \ell_\infty$): The 'Projected Gradient Method' extends the FGM attack to include a projection on the ε-sphere, ensuring that generated samples do not fall outside of the feasible space [35]. This method is iterative, and was restricted here to ten such iterations. The imposed feasibility conditions on the FGM attack were in $\varepsilon \in \{.001, .01, .03, .1, .2, .3, .5, .8, 1\}$.

Deep (ℓ_2): the 'Deepfool Attack' [43] finds the minimal separating hyperplane between two classes and then adds a specified amount of perturbation to ensure it crosses the boundary by using an approximation of the model gradient by approximating the n most likely class gradients where $n \in \{1, 3, 5, 10\}$, speeding up computation by ignoring unlikely classes [43]. This method is iterative and was restricted here to ten such iterations.

Pixel (ℓ_0): the 'PixelAttack' uses a well-known multi-objective search algorithm [28], but tries to maximise false confidence while minimising the number of perturbed pixels. This method is iterative and was restricted here to ten such iterations. For ε, we tested $\{1, 4, 16, 64, 256\}$ pixels.

Thresh (ℓ_∞): the 'Threshold' attack also uses the same multi-objective search algorithm as Pixel to optimise the attack, but tries to maximise false confidence using a penalty term on the loss function while minimising the ℓ_2 perturbation distance. This method is iterative and was restricted here to ten such iterations. We tested penalty terms corresponding to $\{1, 4, 16, 64, 256\}$

HSJ (ℓ_2, *queries*): the 'HopSkipJump' attack, in contrast to the attacks above, does not need access to model gradients nor soft class labels, instead relying on an offline approximation of the gradient using the model's decision boundaries. In this case, the strength is denoted by the number of queries necessary to find an adversarial counterexample [12]. This method is iterative and was restricted here to ten such iterations.

5.5 Survival Models

The exponential, Weibull, log-normal, log-logistic, and generalised gamma AFT models were tested as well as the Cox proportional hazards model using the `lifelines` [16] package in Python. For each attack, the attack-specific distance metric (or pseudo-metric) outlined in Section 5.4 was identified. To compare the effect of this measure against other attacks, the values were min-max scaled so that all values fell on the interval $[0, 1]$. The same scaling was done for the defences. Because this strength parameter isn't directly comparable across attacks or defences, a dummy variable was introduced for each attack and defence, allowing an estimate of their effect relative to the baseline hazard. The number of epochs and the number of layers were tracked for the models, as well as the training and inference times. The metrics outlined in Section 3.3 were used for choosing the best-fit AFT model, as is best practice [7].

6 Results and Discussion

Through tens of thousands experiments across many signal-processing techniques (*i.e.*, defences), random states, learning rates, model architectures, and attack configurations, we show that model defences generally fail to outperform the undefended model in either the benign or adversarial contexts—regardless of configuration. Also, that the adversarial failure rate gains of larger ResNet configurations are driven by response time rather than true robustness; that these gains are dwarfed by the increase in training time; and that AFT models are a powerful tool for comparing model architectures and examining the effects of covariates. In the section below, we display and discuss the results for the CIFAR100, CIFAR10, and MNIST datasets for all attacks and defences.

6.1 AFT Models

Table 1 contains the performance of each of these models on the CIFAR10 dataset. For all datasets, the results are roughly comparable with regards to Concordance, but the log-logistic and exponential models marginally outperforms the other models when measured with AIC/BIC. Concordance is identical for both the test and train sets, with gamma and exponential falling behind the others. However, the ICI and E50 across the test train sets is superior for the Weibull, so that model was used to calculate the expected survival time in Figure 2, which is discussed below. Figure 1 clearly shows that more hidden layers do increase the survival time. However, that seems to be driven more by the model query time (see t_{predict} in Figure 1) than inherent robustness (see *Layers* in Figure 1).

6.2 Effect of Covariates

Figure 1 depicts the effect of all attacks, defences, and model configurations on the survival time and Figure 1 depicts the effect of the covariates. Figure 1 clearly demonstrates that increasing the depth of the model architecture does little for adversarial robustness while universally increasing the training time. Furthermore, it reveals something surprising—that increasing the number of epochs tends to increase the failure rate—even across model architectures and all defences. Certain defences can outperform the control model—at the cost of expensive tuning—evidenced by the large variance in performance (see Figure 2). Additionally, we see that an increase in accuracy tends to correspond to a decrease in survival time, confirming the inverse relationship noted by many researchers [10, 6, 42]. As the training time increases, however, the variance of attack times decreases, likely due to the corresponding increase in inference time (see Figure 1, covariate t_{predict}) rather than inherent robustness (see covariate 'Layers'). We formalise this analysis in the next subsection.

6.3 Failures and Cost

Figure 2 depicts the cost-failure ratio (see Equation 9) in both the benign (left) figure and adversarial cases (middle and right figures), using the Weibull model to calculate $\mathbb{E}[T]$. Counter-intuitively, we see that the smallest model (ResNet18) tends to outperform both larger models (ResNet50 and ResNet152). Furthermore, we see that defence tuning is about as important as choosing the right type of defence (see left side of Figure 2), with all defences falling within the normal ranges of each other. However, adding noise to the model output (Gauss-out) tends to underperform relative to the control for all models (see left side of Figure 2). Likewise, the efficacy of a defence depends as much on model architecture as it does on hyperparameter tuning as demonstrated by the large variance in Figure 2. Furthermore, performance across all attacks is remarkably consistent with intra-class variation being smaller than inter-class variation almost universally across defences and model configurations. Finally—and most importantly—we see that every single tested configuration performs incredibly poorly against FGM and PGD.

7 Considerations

The proposed survival and cost analysis has some limitations that we have taken all efforts to minimise and/or mitigate. In order to minimise timing jitter, we measured the process time

	AIC	BIC	Concordance	Test Concordance	ICI	Test ICI	E50	Test E50
Cox	–	–	0.92	0.92	0.07	–	0.05	–
Gamma	–	–	0.51	0.52	0.26	0.17	0.17	0.24
Weibull	9.05e+04	9.05e+04	0.92	0.92	0.02	0.02	0	0.01
Exponential	7.93e+04	7.93e+04	0.86	0.86	0.04	0.19	0.01	0.02
Log Logistic	9.79e+04	9.79e+04	0.92	0.92	0.07	0.08	0.01	0.01
Log Normal	1.14e+05	1.14e+05	0.91	0.91	0.15	0.26	0.08	0.19

TABLE 1. This table depicts the performance metrics (see Section 3.3) for various survival analysis models (see Section 3) according to the methodology described in Section 5. The concordance measures the agreement between a cubic spline fit to the observed data (see: Section 3.3) and the fitted AFT model, with a value of one indicating perfect performance. The ICI score measures the total error between the calibration curve and the fitted model and the E50 refers to the difference between the cubic spline and fitted model at the median of the cubic spline. AIC/BIC respectively refer to the Akaike and Bayesian information criteria which favour smaller scores. Columns including the word test indicate the scores on test data, otherwise it is scored on the training set.

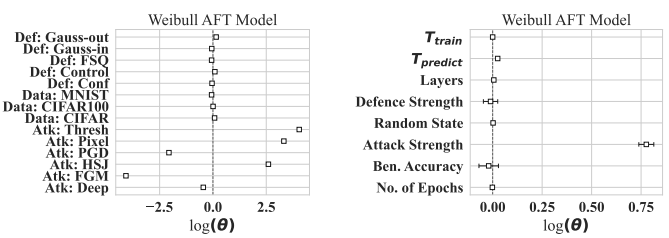


FIGURE 1. The coefficients represent the log scale effect of the dummy variables for dataset (Data), attack (Atk), and defence (Def) on the survival time, with a positive value indicating an increase in the survival time. The right plot depicts the covariates and the left plot depicts the dummy variables for the different attacks, defences, and datasets.

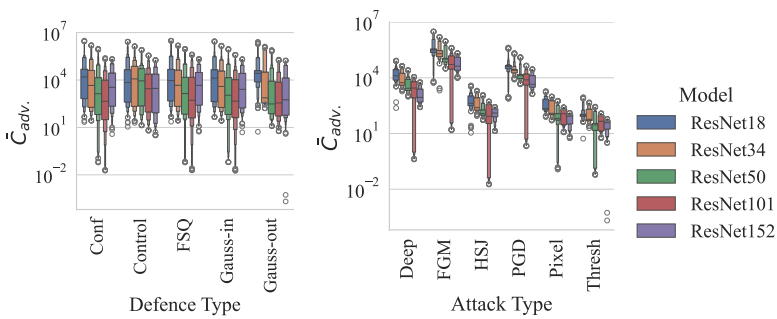


FIGURE 2. This figure depicts the TRASH metric that reflects the ratio of training-to-attack times, where a value $\gg 1$ indicates an essential advantage for the attacker. The violin plots reflect the 95% confidence intervals for each tuned hyperparameter combination. Outliers are indicated with a circle.

for a batch of samples and then assumed that the time per sample was the measured processor time divided by the number of samples. In order to examine a variety of different optimisation criteria for adversarial perturbations, we included several different attacks (see Section 5.4)—though the choice of attack is highly contextual. We must also note that none of these attacks are run-time optimal and are, at best, an underestimate of the true adversarial failure rate [42]. Likewise, testing all known defences would be computationally infeasible. As such, we focused only on the pre- and post-processing technique. While every configuration of ResNet met the bare minimum requirement outlined in Equation 9, real training processes require many thousands of samples and attacks are consistently successful with only one hundred samples. Together, these considerations raise serious concerns about the efficacy of any of these models and defences in the presence of these simple adversaries, meaning attacks will likely be many orders of magnitude cheaper than defences for tested configurations of this model. Furthermore, state of-the art leaderboards[4][5] show that a 99% generalised adversarial accuracy is, at best, optimistic.

8 Conclusion

Convolutional neural networks have shown to be widely applicable to a large number of fields when large amounts of labelled data are available. By examining the role of the attacks, defences, and model depth in the context of adversarial failure rate, this paper presents a reliable and effective modelling framework that applies AFT models to deep neural networks. The metrics outlined Table 1 and explained in Section 3.3 show that this method is both effective and data-agnostic. We use this model to demonstrate the efficacy of various attack- and defence-tuning techniques, to explore the relationships between accuracy and adversarial robustness (Figure 1), and show that various model defences are ineffective on average and marginally better than the control at best. By measuring the cost-normalised failure rate or TRASH score (see Section 4 and Figure 2), it is clear that all tested configurations of ResNet fail to meet the TRASH criterion. The methods can easily extend to any other arbitrary collection of model pre-processing, training, tuning, attack and/or deployment parameters. In short, AFTs provide a rigorous way to compare not only the relative robustness of a model, but of its cost effectiveness in response to an attacker. The measurements rigorously demonstrate that the depth of a ResNet architecture does little to guarantee robustness while the community trends towards larger models [18].

While the train-test split methodology relies on an ever-larger number of samples to increase precision, the survival time method is able to precisely and accurately compare models using only a small number of samples [48, 30] relative to the many billions of samples required of the train/test split methodology and safety-critical standards [39, 37, 38, 42]. In short, by generating worst-case examples (*e.g.*, adversarial ones), one can test and compare arbitrarily complex models *before* they leave the lab, drive a car, predict the presence of cancer, or pilot a drone.

Acknowledgements

Financial support has been provided in part by the Knut and Alice Wallenberg Foundation grant number 2019.0352 and by the eSSENCE Programme under the Swedish Government's Strategic Research Initiative. This material is based upon work supported by the Google Cloud Research Credits program with the award 42030cd8-057f-4365-aed8-1b1afa30c3ee.

References

[1] T. Akiba et al. "Optuna: A next-generation hyperparameter optimization framework". In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 2623–2631.

[2] D. Arp et al. "Dos and Don'ts of Machine Learning in Computer Security". In: *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 3971–3988.

[3] P. C. Austin, F. E. Harrell Jr, and D. van Klaveren. "Graphical calibration curves and the integrated calibration index (ICI) for survival models". In: *Statistics in Medicine* 39.21 (2020), pp. 2714–2742.

[4] A. Bailly et al. "Effects of Dataset Size and Interactions on the Prediction Performance of Logistic Regression and Deep Learning Models". In: *Computer Methods and Programs in Biomedicine* 213 (Oct. 2021), p. 106504.

[5] B. Biggio, B. Nelson, and P. Laskov. "Poisoning attacks against support vector machines". In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. ICML'12. Edinburgh, Scotland: Omnipress, 2012, 1467–1474.

[6] B. Biggio et al. "Evasion Attacks against Machine Learning at Test Time". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, 387–402.

[4] Madry's MNIST Challenge
[5] Croce's Robust Bench

[7] M. J. Bradburn et al. “Survival analysis part II: multivariate data analysis–an introduction to concepts and methods”. In: *British journal of cancer* 89.3 (2003), pp. 431–436.

[8] T. B. Brown et al. “Adversarial Patch”. In: *CoRR* abs/1712.09665 (2017). arXiv: 1712.09665.

[9] J. Byun, H. Go, and C. Kim. “On the effectiveness of small input noise for defending against query-based black-box attacks”. In: *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 2022, pp. 3051–3060.

[10] N. Carlini and D. Wagner. “Towards Evaluating the Robustness of Neural Networks”. In: *2017 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2017, pp. 39–57.

[11] A. Chakraborty et al. “Adversarial Attacks and Defences: A Survey”. In: *arXiv:1810.00069 [cs, stat]* (2018).

[12] J. Chen, M. I. Jordan, and M. J. Wainwright. “HopSkipJumpAttack: A query-efficient decision-based attack”. In: *IEEE symposium on security and privacy (sp)*. IEEE. 2020, pp. 1277–1294.

[13] L. Chen et al. “Lie to Me: A Soft Threshold Defense Method for Adversarial Examples of Remote Sensing Images”. In: *IEEE Geoscience and Remote Sensing Letters* (2021), pp. 1–5.

[14] C. A. Choquette-Choo et al. “Label-only membership inference attacks”. In: *International conference on machine learning*. PMLR. 2021, pp. 1964–1974.

[15] D. Collett. “Modelling survival data”. In: *Modelling survival data in medical research*. Springer, 2015.

[16] C. Davidson-Pilon. “lifelines: survival analysis in Python”. In: *Journal of Open Source Software* 4.40 (2019), p. 1317.

[17] L. Deng. “The mnist database of handwritten digit images for machine learning research”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[18] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo. “Compute and energy consumption trends in deep learning inference”. In: *arXiv:2109.05472* (2021).

[19] DVC Authors. *DVC–Data Version Control*. Github. 2023.

[20] B. Erickson et al. “Machine Learning for Medical Imaging”. In: *RadioGraphics* 37 (Feb. 2017), p. 160130.

[21] D. J. Fremont et al. “Formal scenario-based testing of autonomous vehicles: From simulation to the real world”. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2020, pp. 1–8.

[22] I. J. Goodfellow, J. Shlens, and C. Szegedy. “Explaining and harnessing adversarial examples”. In: *arXiv:1412.6572* (2014).

[23] K. He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[24] W. Huang et al. “Autonomous vehicles testing methods review”. In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2016, pp. 163–168.

[25] P. Kamal. “A study on the security of password hashing based on gpu based, password cracking using high-performance cloud computing”. MA thesis. St. Cloud State, Minnesota, USA, 2017.

[26] D. G. Kleinbaum and M. Klein. *Survival analysis a self-learning text*. Springer, 1996.

[27] A. M. Koay et al. “Machine learning in industrial control system (ICS) security: current landscape, opportunities and challenges”. In: *Journal of Intelligent Information Systems* 60.2 (2023), pp. 377–405.

[28] S. Kotyan and D. V. Vargas. “Adversarial robustness assessment”. In: *PloS one* 17.4 (2022).

[29] A. Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. In: *University of Toronto* (May 2012).

[30] J. M. Lachin. “Introduction to sample size determination and power analysis for clinical trials”. In: *Controlled clinical trials* 2.2 (1981), pp. 93–113.

[31] J. Lawless, J. Hu, and J. Cao. “Methods for the estimation of failure distributions and rates from automobile warranty data”. In: *Lifetime Data Analysis* 1 (1995), pp. 227–240.

[32] G. Leurent and T. Peyrin. “SHA-1 is a shambles: First Chosen-Prefix collision on SHA-1 and application to the PGP web of trust”. In: *29th USENIX*. 2020, pp. 1839–1856.

[33] Z. Li and Y. Zhang. “Membership leakage in label-only exposures”. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2021, pp. 880–895.

[34] Q. Liu, A. Ismail, and W. Jung. “Development of Accelerated Failure-free Test Method for Automotive Alternator Magnet”. In: *Journal of the Society of Korea Industrial and Systems Engineering* 36.4 (2013), pp. 92–99.

[35] A. Madry et al. “Towards deep learning models resistant to adversarial attacks”. In: *arXiv:1706.06083* (2017).

[36] A. Maheshwari, N. Davendralingam, and D. A. DeLaurentis. “A comparative study of machine learning techniques for aviation applications”. In: *2018 Aviation Technology, Integration, and Operations Conference*. 2018, pp. 39–80.

[37] International Electrotechnical Commission. *IEC 61508 Safety and Functional Safety*. 2nd. International Electrotechnical Commission, 2010.

[38] International Electrotechnical Commission. *IEC 62304 Medical Device Software - Software Life Cycle Processes*. 2nd. International Electrotechnical Commission, 2006.

[39] International Standards Organization. *ISO 26262-1:2011, Road vehicles — Functional safety*. https://www.iso.org/standard/43464.html. 2018.

[40] M. H. Meng et al. “Adversarial robustness of deep neural networks: A survey from a formal verification perspective”. In: *IEEE Transactions on Dependable and Secure Computing* (2022).

[41] D. Mery et al. “Modern Computer Vision Techniques for X-Ray Testing in Baggage Inspection”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47.4 (2017), pp. 682–692.

[42] C. Meyers, T. Löfstedt, and E. Elmroth. “Safety-critical computer vision”. In: *Springer Artificial Intelligence Review* (2023).

[43] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. “Deepfool: a simple and accurate method to fool deep neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2574–2582.

[44] M.-I. Nicolae et al. “Adversarial Robustness Toolbox v1.2.0”. In: *CoRR* 1807.01069 (2018).

[45] T. Orekondy, B. Schiele, and M. Fritz. “Knockoff nets: Stealing functionality of black-box models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4954–4963.

[46] A. Saha, A. Subramanya, and H. Pirsiavash. “Hidden trigger backdoor attacks”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 07. 2020, pp. 11957–11965.

[47] P. M. Santos et al. “Universal adversarial attacks on neural networks for power allocation in a massive MIMO system”. In: *IEEE Wireless Communications Letters* 11.1 (2021), pp. 67–71.

[48] C. Schmoor, W. Sauerbrei, and M. Schumacher. “Sample size considerations for the evaluation of prognostic factors in survival analysis”. In: *Statistics in medicine* 19.4 (2000), pp. 441–452.

[49] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, Cambridge, UK., 2014.

[50] K. Simonyan and A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[51] P. Stoica and Y. Selen. “Model-order selection: a review of information criterion rules”. In: *IEEE Signal Processing Magazine* 21.4 (2004), pp. 36–47.

[52] H. Uno et al. “On the C-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data”. In: *Statistics in medicine* 30.10 (2011), pp. 1105–1117.

[53] W. Xu, D. Evans, and Y. Qi. “Feature squeezing: Detecting adversarial examples in deep neural networks”. In: *arXiv:1704.01155* (2017).

[54] O. Yadan. *Hydra – A framework for elegantly configuring complex applications*. Github. 2019.

[55] V. Zantedeschi, M.-I. Nicolae, and A. Rawat. “Efficient Defenses Against Adversarial Attacks”. In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. AISec ’17. New York, NY, USA: Association for Computing Machinery, 2017, 39–49.

Paper IV

A Cost-Aware Approach to Adversarial Robustness in Neural Networks.

Charles Meyers, Mohammad Reza Saleh Sedghpour, Tommy Löfstedt, and Erik Elmroth

Submitted for publication September 2024.

Can we extend the analysis in Paper III to examine the relationship between cost and robustness?

A Cost-Aware Approach to Adversarial Robustness in Neural Networks

Charles Meyers[1*], Mohammad Reza Saleh Sedghpour[2,1], Erik Elmroth[1,2], Tommy Löfstedt[1]

[1*]Department of Computing Science, Umeå University, Umeå, 901 87, Sweden.
[2]Elastisys AB, Sweden.

*Corresponding author(s). E-mail(s): cmeyers@cs.umu.se;
Contributing authors: msaleh@cs.umu.se; elmroth@cs.umu.se; tommy@cs.umu.se;

Abstract

Considering the growing prominence of production-level AI and the threat of adversarial attacks that can evade a model at run-time, evaluating the robustness of models to these evasion attacks is of critical importance. Additionally, testing model changes likely means deploying the models to (*e.g.* a car or a medical imaging device), or a drone to see how it affects performance, making un-tested changes a public problem that reduces development speed, increases cost of development, and makes it difficult (if not impossible) to parse cause from effect. In this work, we used survival analysis as a cloud-native, time-efficient and precise method for predicting model performance in the presence of adversarial noise. For neural networks in particular, the relationships between the learning rate, batch size, training time, convergence time, and deployment cost are highly complex, so researchers generally rely on benchmark datasets to assess the ability of a model to generalize beyond the training data. However, in practice, this means that each model configuration needs to be evaluated against real-world deployment samples which can be prohibitively expensive or time-consuming to collect — especially when other parts of the software or hardware stack are developed in parallel. To address this, we propose using accelerated failure time models to measure the effect of hardware choice, batch size, number of epochs, and test-set accuracy by using adversarial attacks to induce failures on a reference model architecture before deploying the model to the real world. We evaluate several GPU types and use the Tree Parzen Estimator to maximize model robustness and minimize model run-time simultaneously. This provides a way to evaluate

the model and optimise it in a single step, while simultaneously allowing us to model the effect of model parameters on training time, prediction time, and accuracy. Using this technique, we demonstrate that newer, more-powerful hardware does decrease the training time, but with a monetary and power cost that far outpaces the marginal gains in accuracy.

Keywords: artificial intelligence, machine learning, adversarial AI, optimisation, compliance

1 Introduction

1.1 Motivation

Recently, machine learning (ML) using deep neural networks has become a popular way to classify large amounts of data — with applications ranging from medical imaging [1] to aviation [2] and from security [3–5] to self-driving cars [6]. Statistical learning theory [7, 8] provides us no guarantees about the generalization performance of deep neural networks due to the massive number of tunable parameters. To overcome this, neural networks need large amounts of data [9, 10] to train ever-larger model [9], which has yielded increasingly marginal gains on test-set accuracy [11]. It is also clear that reaching safety-critical standards using test-set accuracy would require an infeasibly large test set [12]. Therefore, we propose using Accelerated Failure Time (AFT) models to simulate edge-cases and verify models using a small number of samples. Modern neural networks are massive — they have have grown to be one of the largest consumers of data-center power, especially with the rise of generative AI [13]. For image systems, AlexNet [14] is now considered small with *only* 60 million parameters. ResNet152 has even more parameters at 116 million [15]. However, modern architectures have lead to an explosion in model size with the recent Mamba model boasting a massive 8 *billion* tunable parameters [16]. At these scales, we have no statistical guarantees about the performance of these models and test-set validation would require many, many billions of samples for each and every model change. Furthermore, even if we ignore the immense cost of training modern neural networks, the required number of test samples creates serious questions about the efficacy of the typical train/test split methodology for assessing model generalization [12] since regulatory standards around safety-critical software applications [17–20] clearly define the maximum failure rate to be in the range $[10^{-12}, 10^{-15}]$ depending on the number of lives at risk. Furthermore, ensuring the robustness of ML models against adversarial noise has become a critical concern since inducing a failure at run-time has consistently shown to be trivial [21–26]. Collecting, labelling, and testing a new set of data for every software change — as required by law in most of the world [17–20] — would be prohibitively expensive for these large models. Therefore, a new evaluation methodology is required. We propose to use survival analysis as a methodological framework to model the failure conditions of a machine learning (ML) model. Instead of having a test set large enough to cover all of the failure cases, we generate adversarial samples crafted specifically to make the model fail and then use survival analysis to predict these failures in general. In this work, we

used accelerated failure-time methods to predict model performance as a function of various model parameters. The methodology outlined in Section 3 allows the model-builder to minimize the training cost, optimise for adversarial robustness, estimate the effects of covariates on model performance during routine training procedures. Then, one can perform a cost analysis (Section 4). We then demonstrate the efficacy of this methodology in Section 6 by examining the role of hardware on model performance, the particulars of which are outlined in Section 5.

1.2 Contributions

To tackle the problems of minimizing deployment cost while maximizing the model performance on both the test set and in the presence of adversarial noise we present a scalable and effective methodology (see Figure 1) and software framework (see Figure 2) for the training (see Section 3) and evaluation (see Section 4) of ML models that:

- Optimises for benign and adversarial accuracy simultaneously,
- Demonstrates a scalable and effective method to train a model while simultaneously estimating the effect of various hyper-parameters, and
- Measures the power and monetary cost of deploying a model across different hardware architectures to model the trade-offs between deployment hardware and robustness.
- Demonstrates that, even when we separately measure the effect of the slower clock speed of "inference only hardware" compared to hardware tailored for training, that models trained using the "inference only" devices are more robust for image classification tasks than models built on hardware designed for training.

Section 2 defines the terminology used throughout the paper. Section 3 outlines the training and evaluation methodology presented in this paper. Section 4 discusses the resulting cost analysis framework, arising from the methodology in the previous section. Section 5 outlines the software components and specific experiments that were conducted, while Section 6 contains the results and discussions of those experiments. Section 7 and Section 8 reveal the caveats and the conclusions respectively.

2 Background

In this paper, we evaluate a comprehensive methodology for evaluating model robustness during training time and model the asymptotic effect of the various parameters in the hyper-parameter search space. For the sake of the reader, we provide a section for definitions and requisite background information below.

2.1 Cloud Architectures

ML pipelines play a crucial role in the development and deployment of robust and accurate models. However, managing complex pipelines across diverse central processing unit (CPU) and graphics processing unit (GPU) architectures, ensuring robustness against adversarial attacks, and understanding the relationship between computational

cost, model loss, and prediction accuracy remain ongoing challenges. One popular solution is using a network of interconnected services [27–30] where a *service* is the smallest component of a "cloud-native" software stack. In the context of ML, that might be some software component meant for training, inference, pre-processing, sampling, or any other arbitrarily small part of the data pipeline. A service *mesh* typically consists of a set of interconnected components or proxies deployed alongside the services within the system. These components facilitate various capabilities and functionalities essential for managing the communication between services. Kubernetes has become one of the largest open source projects on the code-sharing website Github [31] and provides a framework for managing, monitoring, and networking a self-scaling set of tools across arbitrary software and hardware architectures using a service mesh. For ML applications, these services are often divided into "training" and "inference" configurations that often have distinct hardware and software configurations [32]. In this work, we leveraged Kubernetes [33] to manage a multi-stage ML pipeline (see Figure 1) and measure the power and cost of training and evaluating the pipeline (see Figure 2) across a variety of different hardware architectures (see Table 1).

2.2 ML Pipelines

ML pipelines are often long-running and complex software tool-chains with many tunable hyper-parameters. Managing, tracking, and controlling for various parameters is non trivial, but many management tools are available [33–35]. In general, a dataset is split into *training* and *test* sets. The test set is then used to determine the best configuration of a given model architecture on a given hardware architecture with the expectation that it will generalize both on the withheld test set and on new data generated and submitted by users. Since different hardware devices have differing amounts of video random access memory (VRAM), the resulting models will have differing optimal configurations if both robustness and training cost are considered. To verify the training process, the test set is validated against the *inference* configuration of a model which may run on different hardware than the *training* configuration to reduce cost, latency, or power consumption. Likewise, Nvidia offers hardware that is marketed as either for training (*e.g.*, v100 and p100) or only for inference (*e.g.*, l4).

2.3 Classifiers

We consider ML classifiers, $K(x;\theta)$ with model parameters, θ. The true labels are denoted by y, and the model predictions by $\hat{y} = K(x;\theta)$ where x is a mini-batch of size N of data samples. The loss function, the measure of discrepancy between the true label and the predicted label, is denoted by $L(y,\hat{y})$. Because of the complexity of modern ML models, researchers rely on numerical optimisation, and one popular choice of optimisation algorithm is *stochastic gradient descent*, which updates the model parameters by taking a step in the negative gradient direction, where the gradient is computed using a subset (a mini-batch) of the training samples instead of all training samples. This procedure is repeated for some number of *epochs* (iterations through

the entire training set). Every iteration takes a step,

$$\theta^{(i+1)} = \theta^{(i)} - \eta \nabla_{\theta^{(i)}} L(y, K(x, \theta^{(i)})) \tag{1}$$

where the gradient is approximated using b samples per mini-batch and η is the *learning rate* (or step size) that is tuned to the particular model and data.

2.4 Learning Rate Selection

Choosing a good learning rate is critical for model performance — greatly effecting both the accuracy and the training cost. A small learning rate will more accurately approximate the class boundaries [36], but will converge slower, all other things being equal. Of course "small" is arbitrarily defined, but the scale of the optimal learning rate will vary with *e.g.*, the mini-batch size, number of epochs, dimensionality of the input data, *etc.* [37]. GPUs with larger amounts of VRAM are able to hold more data in memory at a time, increasing the effective mini-batch size, and reducing the number of model-tuning steps per epoch. A "good" learning rate will allow a model to converge quickly [37, 38] and the ideal mini-batch size will be determined by the memory bandwidth of the GPU and the size of the model and the size and dimensionality of the data. However, since cloud infrastructure is virtualized and shared with other users, the available bandwidth will not necessarily match the peak available bandwidth specified by the manufacturer [39], so researchers must evaluate this *in situ*. Furthermore, since hardware is typically billed by the hour on public clouds, optimising the training and inference times allows a model-builder to minimize the cost of deployment. To optimize for training time as well as robustness (both λ_{ben} and λ_{adv}), the TPE optimization was used (see: Section 3.3). However, before progressing any further, it is first necessary to define robustness.

2.5 Adversarial Attacks

In the context of ML, an adversarial attack refers to deliberate and malicious attempts to manipulate or exploit ML models. Adversarial attacks are designed to deceive or change the model's behavior by introducing carefully crafted input data that can cause the model to make incorrect predictions or otherwise produce undesired outputs. That is, a successful attack is one in which the model outputs on the original, unperturbed data, $\hat{y}$, are not the same as the model outputs on perturbed data, $\hat{y_a}$. That is *adversarial success* or *accelerated failure* is one in which

$$\hat{y} \neq \hat{y}_a = K(x + \varepsilon; \theta), \tag{2}$$

where ε is a noise distance bounded by $0 < \varepsilon \leq \varepsilon^*$ Additionally, one can measure the accuracy of the model when tasked with these adversarial samples, giving us a metric called *adversarial accuracy* which is Eq. 4 calculated on the perturbed samples. The goal of an adversarial attack is often to exploit vulnerabilities in the model's decision-making process or to probe its weaknesses. These attacks can occur during various stages of the ML pipeline, including during training [40, 41], inference [42, 43], or

deployment [21, 22, 24, 42, 44–46]. One of many possible attacks is designed to induce failures as quickly as possible, and is conveniently called the *fast gradient method* [47]. It works by applying noise to a set of samples, x, to generate adversarial examples, x_a, such that,

$$x_a = x + \eta \cdot \text{sign}(\nabla_x L(y, K(x, \theta))). \tag{3}$$

In essence, this is process seeks to increase the loss by changing the input data, in contrast to model training which seeks to minimize the loss by changing the model parameters.

2.6 Adversarial Analysis

In the case of safety- or security-critical domains, considering the worst-case scenario is routine [39]. Whether in the context of automotive safety [6], crytographic systems [48, 49], or healthcare malpractice [1], a component, algorithm, or system is considered broken if the *failure rate* exceeds a certain amount, depending the risk to human life [17]. An order of magnitude more automotive accidents, security breaches, or deaths due to negligence would be unacceptable and, as such, these standards are non-negotiable. However, this would mean testing many millions of samples for every model change that has the potential to injure a human, with orders of magnitude more stringent requirement in the case of potentially fatal systems. This is just not computationally feasible. Instead, researchers can use adversarial failure analysis [12, 22, 50] to improve the precision of our estimates while only using a small set of test-data.

3 Survival Analysis for Robustness Verification during Training

We propose a methodology for model training and verification using accelerated failure-time (AFT) models, drawn from the field of *survival analysis*, with a focus on cost-efficient evaluation methods. AFT models are statistical models used to analyze multivariate effects on the observed failure rate to predict the time-to-failure across a wide variety of circumstances [51, 52]. In medical science, these models are used to make claims like "smokers are twice as likely to die from lung cancer" or used to set the operating limits of manufactured components. This methodology can be used to map the relationship between various model tuning parameters and their effect on model performance. The next section outlines the methodology for modelling the effect of model hyper-parameters during routine training procedures. We precisely outline the methodology for using survival analysis during the training of ML methods.

3.1 Accuracy

Accuracy measures the vulnerability or susceptibility of the model to failures. A larger accurate indicates a higher rate of true classifications, signifying a weaker model in terms of *robustness* against (noise-induced) failures. Throughout, we use the terminology *benign* accuracy to refer to the performance on the test set using unperturbed data and *adversarial* to refer to the performance in the presence of additive noise that

is intended to confuse the model. The subscripts *ben* and *adv* are used respectively. The accuracy, λ, is defined as

$$\lambda := \text{Accuracy} := 1 - \frac{\text{False Classifications}}{\text{Total Classifciations}}, \tag{4}$$

which is generally assumed to indicate the rate of successes in real-world data sampled from the same distribution as the training data [53]. However, the normal test/train split methodology consistently overestimates the model's performance in the presence of adversarial noise [23]. In addition, it ignores the run-time cost of a given architectural decision, caring only about accuracy on benchmark data [9, 10]. Additionally, it has been shown that it is trivial to generate adversarial counter examples that reveal the test/train split methodology to be optimistic at best [12, 21, 22, 24, 40, 42, 46, 54].

3.2 Failure Rate

The failure rate refers to the percentage or proportion of examples that cause the targeted ML model to misclassify or produce incorrect outputs [12]. To encompass the cost of a particular model or attack, the proposed methodology considers failures to be a function over some time interval (*e.g.*, training time, inference time, attack generation time, *etc.*) and some covariates, θ, so that the failure rate is the average time until a failure in a time interval around time, t, such that:

$$h_\theta(t) := \frac{p(\text{False Classification} \mid \theta)}{\Delta t} t,$$

where $p(\text{False Classification} \mid \theta)$ is the probability of a false classification given a particular set of hyper-parameters, θ, Δt is a time interval, and t is a point in time. Note that $1 - \lambda$ is an estimate of this value when $\Delta t = t$ and converges to this value as the number of samples, $N \to \infty$. By modelling accuracy as a function of time, one can then compare the probability of failure to the cost, which is also measured in time, allowing one to make deployment decisions based on the risk analysis standards outlined in IEC61508 [17]. This then allows us to make claims like "increasing training time by X% will yield survival time gains of Y%" [55].

3.3 Optimisation

ML models are typically trained by examining the effect of the entire hyper-parameter space on the resulting accuracy. However, the number of hyper-parameter combinations are often infinite or at least exponential in the number of hyper-parameters, making it infeasible to exhaustively evaluate the entire hyper-parameter search space. Additionally, the goals of test-set accuracy and adversarial accuracy are often at odds [22] — with several researchers noting an inverse relationship between model accuracy and model robustness [12, 22, 54]. Therefore, a proper search would keep this dual-objective in mind. In addition, we attempt to minimize the training time for each piece of hardware since optimal batch size and, therefore, learning rate will be determined by the VRAM, the size bit depth of the data as well as the size and

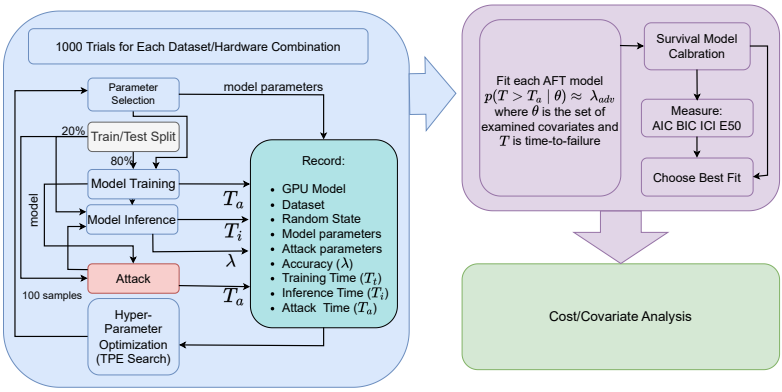


Fig. 1: For each dataset and hardware combination, a random state parameter was chosen at random to decide the test and train sets. Next, model parameters and attack parameters are chosen at random. After 128 random trials, the TPE algorithm attempts to maximize benign and adversarial accuracy while minimizing training time by tuning the model parameters. The random seed for the data split and the attack parameters are sampled independently from this optimization, which is why they are colored differently. The model tuning (blue-box) is discussed in Section 3.3. After the trials are completed, several AFT models are fit (see Section 3.4) and compared (see Section 3.5) using the process depicted in the purple box. Finally we conduct the cost analysis outlined in Section 4 (green box).

bit depth of the model. To maximise adversarial and benign accuracy simultaneously while minimizing training time, we propose the use of the Tree-structured Parzen Estimator (TPE) because it has been shown to converge over tens or hundreds of trials rather the the 1000s of trials typical of other multi-objective optimisation algorithms like CMAES or NSGA-II [56–58]. In addition, we seek to minimize the training time in order to maximize the number of hyper-parameters that can be examined on a fixed budget. Further discussion of the cost analysis can be found in Section 4.

3.4 AFT Models

AFT models are widely used in industrial, medical, or risk-mitigation contexts [51, 52] to model the effect of covariates on a model's expected time-to-failure (also called survival time). For industrial components, this often means testing the component under a variety of extreme circumstances to induce failures prematurely. For medical applications, this is used to model the effect of demographic characteristics or the efficacy of certain treatments on a given disease. For ML, this means inducing failures during training to measure generalization performance.

The point of this is to model the *survival time*, $S_\theta(t)$, as a function of the time, t, and some set of model parameters, θ such that,

$$S_\theta(t) = p(T > t \mid \theta) = \exp\left(-\int_0^t h_\theta(u)\,du\right)$$

where $p(T > t)$ is the probability that a model has not failed by time t ("survives" beyond time t). The expected survival time is

$$\mathbb{E}_{S_\theta}[T] = \int_0^{t^*} S_\theta(u)\,du, \tag{5}$$

where t^* is the latest time observed in the survival data. However, modelling $S_\theta(t)$ requires a choice in modelling function for S_θ. The Log-Logistic, Log-Normal, and Weibull functions are widely used alternatives [52, 55]. For each trial, one can measure the attack generation time to define the time interval and the accuracy to estimate the number of failures and successes in that time interval.

3.4.1 Survival Time

By using adversarial samples (see Equation 2), failures can be induced. The likelihood of that failure is dependent on the amount of adversarial noise, ε, since adversarial noise is known to *induce* failures. That is, the rate of adversarial success (Equation 2), will increase as ε increases. In terms of AFT models, this can be expressed as the expected lifetime of an adversarial sample when treating ε as a covariate. In the language of accelerated failure time models, this can be expressed in terms of the accelerated failure time assumption [52]

$$S_\theta(t) = S_0\left(\frac{t}{\phi_\theta(x)}\right), \tag{6}$$

where ϕ_θ is the acceleration factor, described by the joint effect of the covariates, such that

$$\phi_\theta(x) = \exp\left(\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n\right),$$

and where $x = (x_0, \ldots, x_n)$ is a vector of covariates and $\theta = (\theta_0, \ldots, \theta_n)$ describe the fitted parameters, and ε is used as one of the covariates. That is, when the adversarial noise level changes, it will also alter the expected survival time of a sample. This assumption means we can evaluate the generalization performance using a small number of adversarial samples with our precision coming from careful time measurements rather than massive test sets [52, 55]. However, in order to find the best AFT model, one must compare the tested models using the techniques outlined below.

3.5 Choosing the best AFT Model

To choose a best-fit from the possible AFT functions, one should prepare the collected metrics and scores and then compare them using *e.g.*, Akaike Information Criterion

(AIC), Bayesian Information Criteria (BIC), or Concordance, as per the best-practices for this methodology [51, 52]. For AIC and BIC, that means choosing the smallest value. Concordance, however, is a number between 0 and 1 that quantifies the degree to which the survival time is explained by the model, where a 1 reflects a perfect explanation [52] and 0.5 reflects random chance. By evaluating $\mathbb{E}_{S_\theta}[T]$ under extreme perturbations, one can test the model and minimize the number of evaluated samples [51, 52] rather than relying on the $> 10^{12}$ samples as required by IEC61508 [17]. While the aforementioned metrics measure goodness-of-fit, one should conduct a *survival probability calibration*, where one fits the AFT model to the data and compares it to a cubic-spline that is meant to capture the un-modelled relationship between failures and time [59]. These plots can be used to visually inspect the goodness-of-fit of various models, as in Figure 7, and this method is called *survival probability calibration* (used in Figure 1). The integrated calibration index (ICI) as well as the error at the 50th percentile E50 [59] can then be calculated. These are the mean absolute difference between observed and predicted probabilities and the median absolute difference between observed and predicted probabilities, respectively [59]. Additionally, the data were split into a training set that was used to fit the model (80%) and an unseen test set (20%) and the concordance, ICI, and E50 were measured for both.

4 Cost Analysis

In addition to the survival analysis, we can use an estimate of survival time to conduct a cost analysis as dictated by IEC61508 [17] which allows us to quantify the marginal risk. To quantify this marginal risk, one must measure the benign and adversarial accuracies (see Equations 2 and 4), the model training time, t_t, the model inference time or latency, t_i, the attack generation time, t_a, the cost per hour for a particular hardware, C, as well as the power consumption, P, of each tested model and attack.

4.1 Accuracy

In order to estimate the number of failures in a given time period, we measured both the benign accuracy and the adversarial accuracy, which reflect the normal test-set accuracy and the test-set accuracy in the presence of additive noise in the direction that maximizes loss. Under the AFT framework above, we can use the adversarial accuracy as a measure of the survival time across a specified time period as outlined in Figure 1.

4.2 Training Time

The training time, T_t, is the time it takes to evaluate n samples when t_t is the training time per sample. It is defined as

$$T_t := t_t \cdot n \cdot m,$$

where m is the number of epochs.

4.3 Latency

Latency is the time it takes to respond to a query. We assume that latency per sample is

$$T_i := t_i \cdot n,$$

which will be driven by the memory bandwidth (measured in bits/second) of a given CPU or GPU and the size [60] and complexity [15] of a given neural network architecture.

4.4 Attack Generation Time

A successful attack is one that induces failure in a model. That is, the expected survival time, $\mathbb{E}_{S_\theta}[T]$, can be thought of as the average time it takes for an attacker to induce a change in the model output. Assuming a uniform distribution, T_a, for n samples, i iterations, and attack time per sample, t_a, as

$$T_a := t_a \cdot n. \tag{7}$$

In reality, the attack time will not be drawn from a uniform distribution and prior research [55] has shown that by treating model and attack parameters as covariates, one can model the survival time accurately by using and AFT model, such that:

$$t'_a \approx \mathbb{E}_{S_\theta}[T] = \int_0^{t^*} S_\theta(t)dt. \tag{8}$$

4.5 Cost

Furthermore, we approach the cost of deployment at two scales. Firstly, we consider the cloud-rental scale, where a small-business might test and deploy a model using *e.g.*, the Google Cloud Platform (GCP) compute costs as a measure of total cost. However, at a certain scale or with certain applications, it is more appropriate to talk about cost in terms of power (*e.g.*, to deploy a self-driving car with a useful operating range). Finally, we define metrics that provide an efficient way to minimize the latency, cost of deployment, and maximize the generalized performance of a model. We define the training cost as

$$C_t = C_h \cdot T_t,$$

the cost of model inference time as,

$$C_i = C_h \cdot T_i,$$

and the cost of an attack as,

$$C_a = C_h \cdot T_a.$$

where C_h is the cost per unit time for the hardware, T_t is the training time, T_i is the inference time, and T_a is the attack generation time, as defined above.

4.6 TRASH Score

With an estimate of the expected survival time in hand, quantify the cost-normalized failure rate, or the ratio of training time to attack time. Assuming that the cost scales linearly – as discussed above – the model builder's cost is proportional to the training time ($C_t \propto t_t$), and the attacker's cost is proportional to the attack time, which is approximately the expected survival time ($C_a \propto t'_a \approx \mathbb{E}_{S\theta}[T]$). Using the definition of ε from Equation 2 and definition of t'_a from Equation 8, we can express the cost of failure in adversarial terms as follows:

$$\text{TRASH} = \frac{t_t}{\mathbb{E}_\theta[T]} = \frac{t_t}{t'_a}. \tag{9}$$

4.6.1 Power

The power consumption for a particular piece of hardware, P_h, measured in Watts (Joules per second), can be thought of similarly such that the total power consumption of model training is

$$P_t = P_h \cdot T_t,$$

the power consumption during model inference is

$$P_i = P_h \cdot T_i,$$

and the power consumption during attack generation is

$$P_a = P_h \cdot T_a,$$

where P_h is the cost per unit time for the hardware, T_t is the training time, T_i is the inference time, and T_a is the attack generation time, as defined above.

5 Experiments

This section outlines specific implementation details for the evaluation of the survival analysis methodology outlined in Section 3 and the cost analysis methodology outlined in Sec 4.

5.1 Cloud Platform and Hardware

To conduct the experiments and have access to different types of hardware, we utilized GCP. Six virtual machines running container optimised operating system provided by GCP constituted the test-bed. Using Google Kubernetes Engine 1.27.3 and Containerd 1.7.0, a cluster consisting of six worker nodes was created. Three worker nodes were responsible for running the monitoring platforms– Prometheus 2.47.2 and Grafana 10.2.0. These nodes were of the "e2-medium" instance type provided by GCP. In total, three GPU architectures were used–the Nvidia P100, V100, and L4. For P100 and V100 GPUs, the "n1-standard-2" type was used for the nodes and for L4 GPUs the "g2-standard-4" was used.

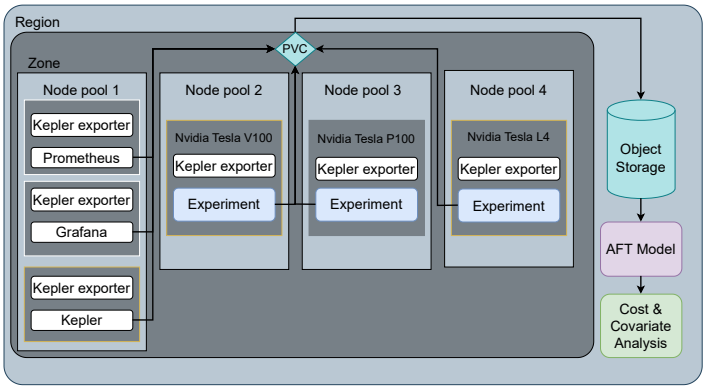


Fig. 2: For our experiments we used four node pools from Google Cloud Platform, each has a particular responsibility. The first node pool includes 3 different nodes responsible for hosting monitoring services such as Prometheus and Grafana. The other node pools each had one node with a specific GPU. The KEPLER exporter is then deployed on each node as DaemonSet to monitor the resource usage. All the storage requirements during the experimentation such as storage for experiments and monitoring data were then stored on storage provided by persistent volume claim (PVC). A PVC is a request for storage by user in Kubernetes which is then connected to the object storage. The blue experiment, blue-green object storage, green analysis component, and purple AFT component correspond to the same colors in Figure 1.

To assess the energy consumption of the experiments, KEPLER deployed on a was used to measure the power consumption of each experiment [61]. This approach enables gathering energy consumption data on granular levels as it runs in Kubernetes cluster and capable of collecting energy consumption of Kubernetes components. In essence, KEPLER uses extended Berkeley Packet Filter to probe energy-related system stats and exports them as Prometheus metrics. This filter can be described as a lightweight and sandboxed virtual machine (VM) in kernel space. The filter programs are invoked by the kernel when certain events occur. Examples of such events include system calls or network activity. These processes enable deep analysis and full control over different events with low overhead [62]. A diagram of the cloud architecture can be found in Figure 2. Finally, for a cost-effective model evaluation technique, all of the experiments were restricted to a single one-thousand United States dollar budget (a research grant from Google). Approximately 10% of this was used for development and 90% was used for the evaluations. Over the course of the evaluations, 6% of the budget went to the storage, 6% to the monitoring, and the remaining 88% went the GPUs.

5.2 AFT Models

For each hardware configuration and dataset, the TPE algorithm was used to select the hyper-parameters [56, 63], and it was iterated for 1000 trials. This was chosen to be substantially larger than the 200 used successfully by Watanabe *et al.* [58] while still allowing for reasonable run times [64]. We used three optimisation criteria: benign accuracy, training time, and adversarial accuracy, seeking to maximize both benign and adversarial accuracy while minimizing training time (and therefore deployment cost). We selected a set of parameters as per the TPE algorithm and trained on 80% of the samples for each of the MNIST, CIFAR10, and CIFAR100 datasets. Of the remaining samples, 100 were withheld to be attacked and used to evaluate the adversarial accuracy. Figure 1 illustrates this methodology. For each dataset, we tested this on ten random splits of the data to create 10 unique test/train pairs. For each trial, we recorded attack generation time, model training time, model inference time, benign accuracy and adversarial accuracy, and the size of the training set, test set, and attack set. Using these values, we fit an AFT model to the number of failures (indicated by accuracy and sample size) and the attack generation time after adding dummy variables for the dataset and hardware device. Additionally, we approximate the AFT model using the methodology depicted in Figure 1.

5.3 Datasets

The AFT models were evaluated using the MNIST [65], CIFAR10 [66], and CIFAR100 [66] datasets, chosen primarily for their standardized use in adversarial analysis [22, 23, 67, 68] and decades of experimental results. Before training, we centered and scaled the data so that the attack distance would be analogous for all tested datasets. Furthermore, to reduce the complexities of system overhead, distributed or federated training, and the effect of shared cloud environments, we restricted ourselves to datasets that were small enough to reside entirely within GPU memory with the model, since the disk read speed in cloud environments is incredibly variable.

5.4 Models

The evaluations were restricted to a single model. Primarily, this was done to meet the budgetary constraints since evaluating more models would mean evaluating fewer pieces of hardware. As discussed in Section 2.4, the relationship between hardware specifications, hyper-parameters, and performance is highly complex and hard to predict. So, we sampled learning rates $\in [10^{-6}, 1]$, batch sizes $\in [1, 10^5]$, and epochs $\in [1, 50]$ for MNIST and CIFAR10 on the P100 and V100. For CIFAR100, the range of the tested epochs was increased to be $\in [1, 100]$. The Feature Squeezing defence [69] was used to evaluate the efficacy of different bit-depths on the L4 hardware, as provided by IBM's adversarial robustness toolbox [26] with bit depths $\in [4, 8, 16, 32, 64]$ which casts the inputs into `pytorch` compatible arrays. Model parameters were chosen using the `optuna` optimisation framework, the configuration was handled by `hydra`, and `dvc` was used to ensure reproducibility and aid in collaborative development.

5.5 Attacks

To examine the effect of model parameters at run-time, evasion attacks, which attack the model at the prediction stage, were examined. Prior research [12, 55] has shown that the Fast Gradient method (see Eq. 3) is consistently the most effective at inducing a large number of failures in a small amount of time. To evaluate the effect of adversarial noise on the samples, the noise levels were varied $0 < \varepsilon \leq 0$. This was done using the `adversarial-robustness-toolbox` package maintained by a team at IBM [26].

5.6 GPU Configurations

Several hardware configurations were tested, that had various hourly costs, peak power demands, and theoretical memory bandwidths. The V100 was chosen as a baseline, since it is routinely used in the literature [70, 71]. The P100 architecture comes from the same line of server-grade GPUs, but from an older generation. The L4, however, is advertised as a machine built for inference — not training — relying on a smaller number of bits per tensor core. Consequently, the number of operations per second depends on the bit depth of the data and model weights, with peak numbers outlined in Table 1 for 8-bit inputs. The rental cost of the hardware, measured in United States Dollars per hour, indicates the operating cost of a given model. To calculate this, the price per hour from each cloud service pricing page [72, 73] were used and the cost of training (C_t) and the cost of inference (C_i) calculated from the cost of hardware (C_h), the training time (T_t), and the inference time (T_i).

	V100	P100	L4
Cost (USD/hour)	2.55	1.60	0.81
Power (Watts)	250	250	72
Memory Bandwidth (GB/s)	900	732	300

Table 1: Hardware specifications for the tested GPUs. The specifications were retrieved from Nvidia's website at the following links: V100 Datasheet, P100 Datasheet, and L4 Datasheet. Prices were retrieved from Google Cloud Platform for the `europe-west4` region on 3 December 2023.

5.7 Survival Analysis

In addition to the optimisation criteria of benign/adversarial accuracy and training time, prediction times, attack times, power consumption, batch size, and number of epochs were also collected to be used as covariates in the AFT model, $S_\theta(t)$. To fit the AFT model and to plot the effect of the covariates, the `lifelines` Python package was used [74]. The Weibull, Log Logistic, and Log Normal AFT models (see Section 3) were also compared.

6 Results and Discussion

This section presents and discusses the results for all of the experiments across all datasets and hardware. First, the accuracy, training time, inference, and monetary cost are discussed in Sections 6.1–6.2, followed by the fitted AFT model in Section 6.3.

6.1 Accuracy

Figure 3 shows the benign (left) and adversarial (right) accuracies for all datasets and hardware. It demonstrates little to no change in accuracy or adversarial accuracy, regardless of hardware. The benign accuracy decreases with difficulty (CIFAR10 vs MNIST) or with the number of classes (CIFAR10 vs CIFAR100). For all three datasets, the adversarial accuracy becomes the reciprocal of the number of classes (*i.e.*, the accuracy we would expect with random data), demonstrating the efficacy of the attack outlined in Section 3.3.

6.2 Time, Power, Cost

Figure 4 reveals the training time, inference time, and attack generation time across all datasets and models. Slower hardware (see Memory Bandwidth in Table 1) is only a few milliseconds slower, so this is unlikely to be noticeable to an end-user during inference. This should be no surprise since the less-capable hardware is meant to be used exclusively for inference. Additionally, the attack time (right sub-figure) increases with both the training (left) and prediction times (center) across hardware and datasets — an expected result since this is driven by the number of samples and the inference time (see Equation 7). We can also see that average training time per sample is takes slightly more than than attack generation time per sample (see Figure 4 left and right plots).

The power consumption during training, inference, and attack generation for all hardware and datasets is illustrated in Figure 5. It tracks monetary cost (Figure 6) closely, probably because that's the predominant cost for the datacenter itself [75], so it is unsurprising that cloud billing is correlated with the power requirement. Furthermore, we see that the largest dataset (CIFAR100) and smallest GPU (L4) require the least amount of power (Figure 5) for prediction, while differences between hardware and datasets are insignificant for the training and attack metrics.

The monetary cost for each dataset and each piece of hardware is shown in Figure 6. For all three datasets across all three pieces of hardware, the cost of training on a single sample often exceeds the cost of attacking a single sample. In the best case scenario, they are comparable, but attacks consistently succeed with only 100 samples (Figure 3) while model training requires orders of magnitude more.

6.3 AFT Models

Table 2 shows the performance metrics for all three AFT models, as outlined in Section 3.4. A total 75% of the available data were used to build the AFT models and 25% were withheld for evaluation. The concordance is both strong (> 0.5) and similar for all three AFT models and both the train and test sets. We observed no more than

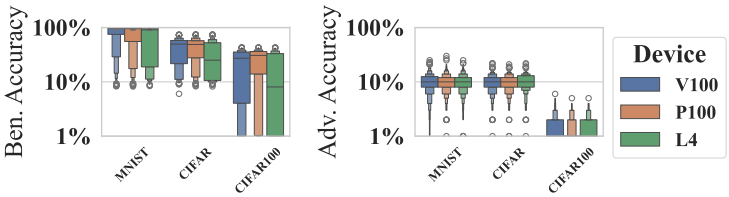


Fig. 3: Benign and adversarial accuracy across all hardware and datasets for all 1000 trials using plots that depict the distribution of the y-axis values using the width of the plot. Each color is a different device and the datasets are displayed along the x-axis. Outliers are denoted with a white dot.

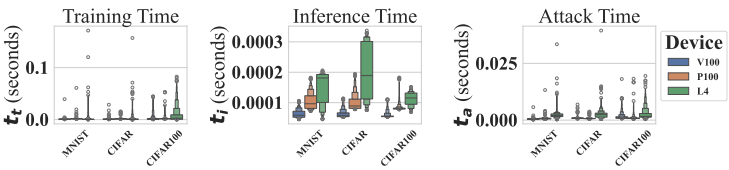


Fig. 4: This depicts the training, inference, and attack times for all hardware and datasets for all 1000 trials using plots that depict the distribution of the second axis values using the width of the plot. The time per sample was assumed to be uniform across the batch of samples for each training, inference, or attack measurement. Each color is a different device and the datasets are displayed along the first axis. Outliers are denoted with a white dot. For these plots, the second axes have been scaled by the number of samples for the sake of comparison.

1% mean absolute error in the probabilities ICI and no error in the median probability E50 across all three AFT models. Figure 7 depicts the observed and predicted probabilities for all three AFT models. All three models have strong predictive power (see Table: 2), with an average test error of around 1% ICI and a similar error for the median sample E50. We also see that these metrics are functionally identical across all AFT models as well as between the test and train set of each model.

Figure 8 shows the log-scale coefficients for the AFT model. It shows that epochs, batch size, and training time have no effect on the survival time. Additionally, the covariate value of 0 for random state indicates that random permutations of the training and test data have no effect on the survival time, which is expected. However, we can clearly see that inference time is as strong an indicator as the attack noise distance, revealing that model speed is nearly as important as the the noise value of the attack

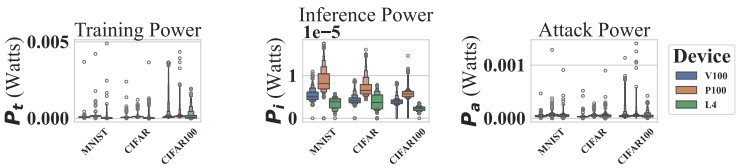


Fig. 5: This depicts the training, inference, and attack times for all hardware and datasets for all 1000 trials using plots that depict the distribution of the second axis values using the width of the plot. The power per sample was assumed to be uniform across the batch of samples for each training, inference, or attack measurement. Each color is a different device and the datasets are displayed along the first axis. Outliers are denoted with a white dot. For these plots, the second axes have been scaled by the number of samples for the sake of comparison.

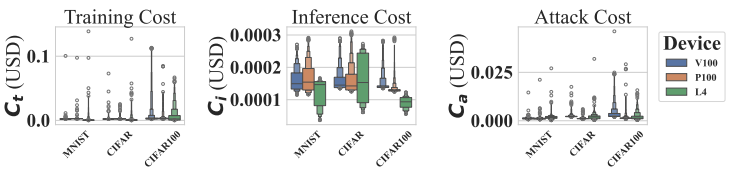


Fig. 6: This depicts the training, inference, and attack times for all hardware and datasets for all 1000 trials using plots that depict the distribution of the second axis values using the width of the plot. The cost per sample was assumed to be uniform across the batch of samples for each training, inference, or attack measurement. Each color is a different device and the datasets are displayed along the first axis. Outliers are denoted with a white dot. For these plots, the second axes have been scaled by the number of samples for the sake of comparison.

(ε). Furthermore, we see that benign accuracy is negatively correlated with the survival time, confirming previous assertions that robustness ($S_\theta(t)$) is inversely related to benign accuracy (which is the standard indicator of generalization performance) [22].

Using the $S_\theta(t)$ described in Section 3, we can model the effect of various hardware devices, which is depicted in Figure 8. It clearly shows that hardware choice has a relatively small effect ($\pm 20\%$) on robustness, despite the much larger disparity in cost outlined in Table 1.

6.4 Why Cost Matters

In security analysis, it is routine to think in optimistic terms for both the attacker and defender. In cryptography, these ideal attack- and defence-scenarios are used to test

Table 2: Comparison of AFT Models across all hardware and datasets. AIC and BIC are measures of goodness-of-fit, with a smaller value being preferred. Concordance (Conc) is a value between 0 and 1 that reflects how what proportion of events (failures) can be explained by the model. ICI measures the average error between a cubic-spline and the model and E50 measures the median error between the spline and the model. These measured on both the train and test sets of the collected data, with the latter being denoted "Test".

	AIC	BIC	Conc	Test Conc	ICI	Test ICI	E50	Test E50
Weibull	$-2.11 \cdot 10^4$	$-2.11 \cdot 10^4$	0.84	0.83	0.00	0.01	0.00	0.00
Log Logistic	$-2.18 \cdot 10^4$	$-2.18 \cdot 10^4$	0.84	0.83	0.01	0.01	0.00	0.00
Log Normal	$-2.25 \cdot 10^4$	$-2.25 \cdot 10^4$	0.84	0.83	0.01	0.00	0.00	0.00

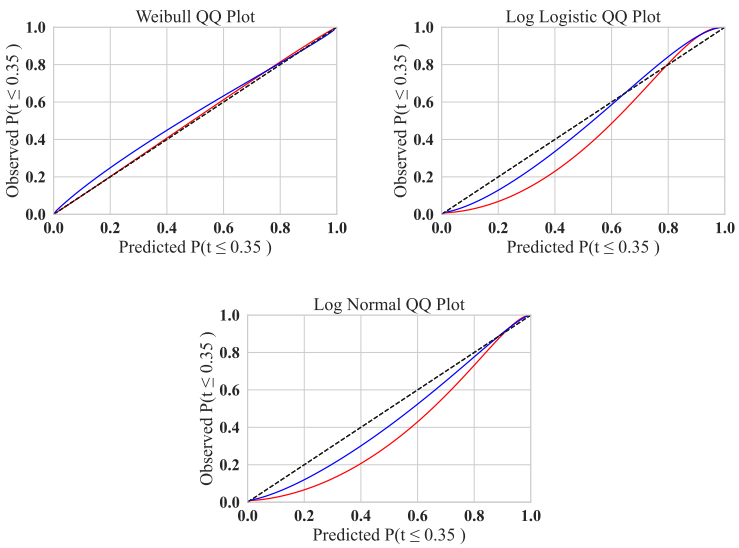


Fig. 7: The red and blue lines depict the relationship between the modelled number of failures (x-axis) and the measured number of failures (y-axis) for the training set (blue) and test set (red). The dotted line indicates a cubic spline fit to $S_\theta(t = t_a) \approx \lambda_{adv}$. A model that fits this cubic spline exactly would be a diagonal line at $y = x$. The process of comparing the fitted AFT model to a default cubic spline is called *survival probability calibration* and is discussed in more detail in Section 3.5.

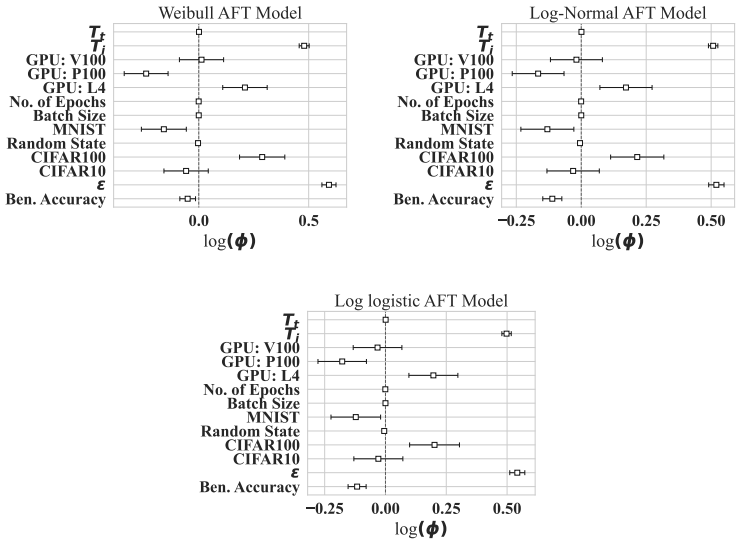


Fig. 8: Coefficients of the Covariates for the Weibull, Log-Normal, and Log-Logistic AFT models. Here, "Random State" is used as a control variable that should be (and is) close to 0. A positive value indicates that covariate increases the survival time and negative value indicates that covariate decreases the survival time. The symbols T_i and T_t refer training and inference time for all samples, while *Ben. Accuracy* refers to the benign accuracy (accuracy on the un-altered samples), and ε is the noise distance.

the computational feasibility of subverting a particular cryptographic system [48, 49] (*e.g.*, whether or not a given cryptographic method should be considered "broken"). Using AFT, one can model the Pareto front — the set of points that are superior to all others in the search space — for all objectives [63]. By using a whitebox attack to generate the failures, we ensure that our defender operates under the worst case scenario while the attacker operates under their best case scenario to yield a worst-case failure rate. As such, we center our analysis on the whitebox Fast Gradient Method (FGM) [47] (see Eq. 3) which is both effective and fast [12]. If the cost to a model builder is much larger than the cost to an attacker, then it is clear that the model is "broken" in this cryptographic sense and can be discarded as ineffective [55]. Figure 9 depicts the TRASH score defined in Equation 9, that quantifies the per-sample ratio of training to attack time. It is clear that even with the reduced GPU bandwidth (Table 1), that the L4 model is not only superior in terms of cost (Table 1), but also in terms of robustness (Table 1), presumably since the extra bit-depth provided by the

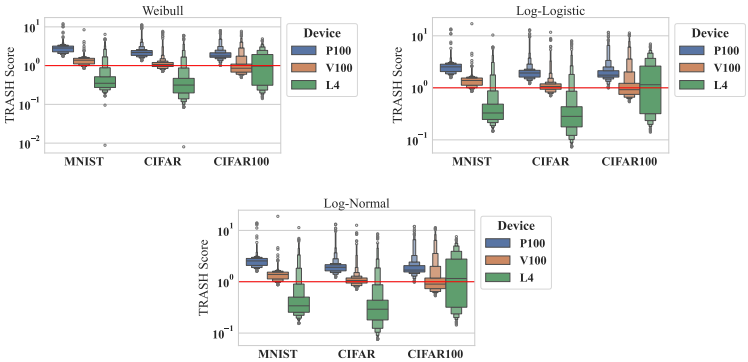


Fig. 9: The *training rate and survival heuristic* score (TRASH score) that depicts the per-sample ratio of training time to attack time for each of the tested AFT models. If this value is greater than one (the red line), then the model can be discarded as ineffective. The x-axis shows each dataset, the y-axis shows the TRASH score and each GPU model is given its own color. Outlier scores are depicted with a white dot.

P100 and V100 ends up being useless noise anyway when operating on 8-bit images like MNIST and CIFAR.

6.5 Advantages of this Methodology

The primary advantage of this methodology over the traditional test/train split measurements is that the precision of the latter is determined by the number of samples; whereas, the precision of the survival time estimate is driven by the resolution of our timing measurements. For test-set accuracy, meeting the weakest safety-critical IEC standard (one failure in a million) would require many millions of samples to confidently conclude that a model is safe. However, the presented AFT model has an average error rate of 1% while requiring a very small number of samples (10 sets of 100) (see Table 2). Because of the small number of samples needed for building AFT models (when compared to testing against massive in-distribution test sets), AFT models could, for example, act as a unit test in ML applications. This is in contrast to a full-system integration tests to evaluate changes to a single model, signal processing technique, data storage format, or API access mechanism [76, 77]. It could also be used to highlight error-prone classes or other subsets of data to reduce error or create synthetic samples as is common in medical research [52]. Furthermore, by isolating changes and testing them as quickly as possible, it is easier to parse cause and effect when compared to full-system integration tests that could include many changes from many different development teams and require live and potentially dangerous systems (like cars, drones, or industrial equipment) to effectively test. To further increase development velocity, these models can quantify the marginal risk associated with each

change, as dictated by the IEC 61508 standard [17], allowing the model builder to make a quantitative assessment about the efficacy of a model change without testing the effect on real users. Additionally, since AFT models have strong predictive power for untested hyper-parameter combinations, this method has a potential to reduce the search space by quickly eliminating candidates that are unlikely to increase the survival time.

7 Considerations

We have taken much care to conduct all timing measurements as carefully as possible. Primarily, to minimize timing jitter and account for GPU parallelization, we assumed that the time-to-failure during the benign and adversarial accuracy measurements was uniform across the samples in each trial. While it is very possible (if not altogether guaranteed) that some classes or samples are easier to attack than others, we assume that this averages out over the 100 samples given to each attack. Further work examining, for example, the effect of imbalanced datasets on the the distribution of survival times across classes is outside the scope of this work, though the methodology would remain identical. Similarly, one could examine survival times for samples near the class boundary and compare them with prototypical samples near the center of the class cluster. However, given the strong results and uniformity across AFT functions in Table 2, the uniform time assumption appears to be insignificant, though accounting for the failure rate per sample or class is likely to account for some of the remaining unexplained variance. Additionally, we chose a model and set of datasets small enough to fit entirely in GPU memory to minimize the confounding factors around parallelized, distributed, and/or federated learning as well as the complexities around storage hardware, file systems, and various access mechanisms. Larger models acting on larger datasets are likely to perform better on hardware with a higher GPU bandwidth, but the 48GB of VRAM provided by the L40 should be more than sufficient for vision tasks based on 8-bit images standard in the literature (*e.g.* MNIST, CIFAR10, CIFAR100). Furthermore, the attack batch size and the training batch size were set to be the same for every trial for each dataset and piece of hardware so that the attack timing would mimic the usage of regular users. If anything, the smaller sample size of the attacks means that proportionally more parallelization overhead is needed per sample, leading to an underestimate of the attack efficacy compared to the benign measurements. Furthermore, while FGM is fast, it is not guaranteed to find failures as quickly as possible, meaning that the survival time estimate is likely to be overestimated. In general, distributed/federated models were out of scope for this work, but architectural choices regarding the configuration of such methods could be evaluated using the cost and survival analysis techniques outlined above. The hyper-parameter search was restricted to only a single evasion attack (see Eq. 3) in the interest of time and budget, though this analysis would generalize to other evasion, extraction, inversion, or poisoning attacks [40, 43, 44, 50].

8 Conclusions

In this work, we applied survival analysis to cloud-native ML, developed a method to quickly and efficiently test model parameter choices and evaluated them in the presence of adversarial noise, allowing one to the model performance as a function of tested model parameters. Using this technique, it is clear that reduced run-times with newer and/or larger hardware, that adversarial robustness is not substantially effected and that the specific choice of attack parameters is far more important than model training time.

The experiments conducted confirm that this methodology is both sound and cost-effective. While 88% of the cloud budget went to GPU rentals, the data show how cutting that cost by 75% and using less power-intensive hardware designed for 8-bit image processing would be *more* effective than using the 32-bit datacenter GPUs (V100 and P100) for typical image classification tasks. Using proved to be a cost-effective as real-time monitoring only used a fraction of the budget (6%). Figure 3 confirms the efficacy of the TPE algorithm discussed in Section 3.3, allowing an infrastructure engineer to ignore the intricacies of learning rate optimization while tuning the model to a particular piece of hardware. Additionally, Table 2 and Figure 7 show that AFT models are an effective way to model the survival time across hardware variants. By comparing the training, inference, and attack times, costs, and power consumption across different hardware devices (Figures 4- 6), it is clear that attacks generally cost less than training, even when we ignore the fact that attacks are consistently effective using a small number of samples while training relies on many thousands (see Figures 1 & 3). This analysis is further confirmed by examining the coefficients of the AFT models in Figure 8 wherein it is evident that neither training time (T_t) nor the number of epochs have noticeable effect on the survival time, but that model latency (T_i) does.

Regardless of which AFT model is chosen, the model suggests that the P100 decreases the survival time compared to the V100 and the L4 increases the survival time (both compared to the V100, which has no significant effect). Furthermore, the coefficients again confirm an inverse relationship between test set (benign) accuracy and the adversarial robustness (*i.e.*, survival time). Finally, even when we account for the reduced GPU bandwidth (see Table 1) and increased training time (see Figure 4), it is clear the the L4 model is superior in terms of robustness and cost-effectiveness (see Figure 9), despite it being advertised as "inference only" by the manufacturer. Furthermore, the data demonstrate the un-advertised training capability of the Nvidia L4 GPU which yields a 20% increase in survival time while costing 75% less than the V100 that's been typical in the literature over the last several years.

8.1 List of Abbreviations

AFT Accelerated Failure Time.
AIC Akaike Information Criterion.

BIC Bayesian Information Criteria.

CPU central processing unit.

E50 error at 50th percentile.

FGM Fast Gradient Method.

GCP Google Cloud Platform.
GPU graphics processing unit.

ICI integrated calibration index.

KEPLER Kubernetes Efficient Power Level Exporter.

ML machine learning.

TPE Tree-structured Parzen Estimator.
TRASH training rate and survival heuristic.

VRAM video random access memory.

9 Declarations

9.1 Ethics Approval and Consent to Participate

Not Applicable

9.2 Consent for Publication

All authors consent for this manuscript to be published under the terms specified by the *Journal of Cloud Computing*.

9.3 Funding

Financial support has been provided in part by the Knut and Alice Wallenberg Foundation grant number 2019.0352 and by the eSSENCE Programme under the Swedish Government's Strategic Research Initiative.

9.4 Acknowledgements

Not Applicable

References

[1] Erickson, B.J., Korfiatis, P., Akkus, Z., Kline, T.L.: Machine learning for medical imaging. Radiographics **37**(2), 505–515 (2017)

[2] Maheshwari, A., Davendralingam, N., DeLaurentis, D.A.: A comparative study of machine learning techniques for aviation applications. In: 2018 Aviation Technology, Integration, and Operations Conference, p. 3980 (2018)

[3] Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., Rieck, K.: Dos and don'ts of machine learning in computer security. In: 31st USENIX Security Symposium (USENIX Security 22), pp. 3971–3988 (2022)

[4] Mery, D., Svec, E., Arias, M., Riffo, V., Saavedra, J.M., Banerjee, S.: Modern computer vision techniques for x-ray testing in baggage inspection. IEEE Transactions on Systems, Man, and Cybernetics: Systems **47**(4), 682–692 (2016)

[5] Travaini, G.V., Pacchioni, F., Bellumore, S., Bosia, M., De Micco, F.: Machine learning and criminal justice: A systematic review of advanced methodology for recidivism risk prediction. International journal of environmental research and public health **19**(17), 10594 (2022)

[6] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al.: End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 (2016)

[7] Vapnik, V.N.: An overview of statistical learning theory. IEEE transactions on neural networks **10**(5), 988–999 (1999)

[8] Shalev-Shwartz, S., Ben-David, S.: Understanding Machine Learning: From Theory to Algorithms. Cambridge university press, Cambridge, UK (2014)

[9] Desislavov, R., Martínez-Plumed, F., Hernández-Orallo, J.: Compute and energy consumption trends in deep learning inference. arXiv:2109.05472 (2021)

[10] Bailly, A., Blanc, C., Francis, É., Guillotin, T., Jamal, F., Wakim, B., Roy, P.: Effects of dataset size and interactions on the prediction performance of logistic regression and deep learning models. Computer Methods and Programs in Biomedicine **213**, 106504 (2022)

[11] Sun, C., Shrivastava, A., Singh, S., Gupta, A.: Revisiting unreasonable effectiveness of data in deep learning era. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 843–852 (2017)

[12] Meyers, C., Löfstedt, T., Elmroth, E.: Safety-critical computer vision: An empirical survey of adversarial evasion attacks and defenses on computer vision systems.

[13] O'Brien, M., Fingerhut, H., Press, T.A.: A.I. tools fueled a 34% spike in Microsoft's water consumption, and one city with its data centers is concerned about the effect on residential supply. Fortune (2023). https://fortune.com/2023/09/09/ai-chatgpt-usage-fuels-spike-in-microsoft-water-consumption/

[14] Alom, M.Z., Taha, T.M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M.S., Van Esesn, B.C., Awwal, A.A.S., Asari, V.K.: The history began from alexnet: A comprehensive survey on deep learning approaches. arXiv preprint arXiv:1803.01164 (2018)

[15] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

[16] Waleffe, R., Byeon, W., Riach, D., Norick, B., Korthikanti, V., Dao, T., Gu, A., Hatamizadeh, A., Singh, S., Narayanan, D., et al.: An empirical study of mamba-based language models. arXiv preprint arXiv:2406.07887 (2024)

[17] Commission, I.E.: IEC 61508 Safety and Functional Safety. International Electrotechnical Commission (2010)

[18] International Standards Organization: ISO 26262-1:2011, Road vehicles — Functional safety. https://www.iso.org/standard/43464.html (2018)

[19] Axelrod, C.W.: Applying lessons from safety-critical systems to security-critical software. In: 2011 IEEE Long Island Systems, Applications and Technology Conference, pp. 1–6 (2011). IEEE

[20] Acharyulu, P.S., Seetharamaiah, P.: A framework for safety automation of safety-critical systems operations. Safety Science **77**, 133–142 (2015)

[21] Brown, T.B., Mané, D., Roy, A., Abadi, M., Gilmer, J.: Adversarial patch. arXiv:1712.09665 (2017)

[22] Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 39–57 (2017)

[23] Croce, F., Hein, M.: Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In: International Conference on Machine Learning, pp. 2206–2216 (2020). PMLR

[24] Chen, J., Jordan, M.I., Wainwright, M.J.: HopSkipJumpAttack: A query-efficient decision-based attack. In: IEEE Symposium on Security and Privacy (SP), pp. 1277–1294 (2020). IEEE

[25] Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., Mukhopadhyay, D.:

Adversarial attacks and defences: A survey. arXiv:1810.00069 (2018)

[26] Nicolae, M.I., Sinn, M., Tran, M.N., Buesser, B., Rawat, A., Wistuba, M., Zantedeschi, V., Baracaldo, N., Chen, B., Ludwig, H., Molloy, I., Edwards, B.: Adversarial robustness toolbox v1.2.0. CoRR **1807.01069** (2018)

[27] Panchal, D., Verma, P., Baran, I., Musgrove, D., Lu, D.: Reusable mlops: Reusable deployment, reusable infrastructure and hot-swappable machine learning models and services. arXiv preprint arXiv:2403.00787 (2024)

[28] Hasselbring, W., Steinacker, G.: Microservice architectures for scalability, agility and reliability in e-commerce. In: 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), pp. 243–246 (2017). IEEE

[29] Zhou, R., Pang, J., Zhang, Q., Wu, C., Jiao, L., Zhong, Y., Li, Z.: Online scheduling algorithm for heterogeneous distributed machine learning jobs. IEEE Transactions on Cloud Computing (2022)

[30] Singh, N., Hamid, Y., Juneja, S., Srivastava, G., Dhiman, G., Gadekallu, T.R., Shah, M.A.: Load balancing and service discovery using docker swarm for microservice based big data applications. Journal of Cloud Computing **12**(1), 4 (2023)

[31] Github: Octoverse Projects. Github.com (2019). https://octoverse.github.com/2018/projects.html

[32] Wang, Y.E., Wei, G.-Y., Brooks, D.: Benchmarking tpu, gpu, and cpu platforms for deep learning. arXiv preprint arXiv:1907.10701 (2019)

[33] Kubernetes: Kubernetes–an open source system for managing containerized applications. Github (2019). https://github.com/kubernetes/kubernetes

[34] dvc.org: DVC- Data Version Control. Github (2023). https://github.com/iterative/dvc.org

[35] Yadan, O.: Hydra - A framework for elegantly configuring complex applications. Github (2019). https://github.com/facebookresearch/hydra

[36] Cao, Y., Gu, Q.: Generalization bounds of stochastic gradient descent for wide and deep neural networks. Advances in neural information processing systems **32** (2019)

[37] Granziol, D., Zohren, S., Roberts, S.: Learning rates as a function of batch size: A random matrix theory approach to neural network training. Journal of Machine Learning Research **23**(173), 1–65 (2022)

[38] Smith, L.N., Topin, N.: Super-convergence: Very fast training of neural networks using large learning rates. In: Artificial Intelligence and Machine Learning for

Multi-domain Operations Applications, vol. 11006, pp. 369–386 (2019). SPIE

[39] Sajid, M., Raza, Z.: Cloud computing: Issues & challenges. In: International Conference on Cloud, Big Data and Trust, vol. 20, pp. 13–15 (2013)

[40] Biggio, B., Nelson, B., Laskov, P.: Poisoning Attacks against Support Vector Machines. arXiv:1206.6389 [cs, stat] (2013)

[41] Saha, A., Subramanya, A., Pirsiavash, H.: Hidden trigger backdoor attacks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 11957–11965 (2020)

[42] Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., Mukhopadhyay, D.: Adversarial attacks and defences: A survey. arXiv:1810.00069 [cs, stat] (2018)

[43] Orekondy, T., Schiele, B., Fritz, M.: Knockoff nets: Stealing functionality of black-box models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4954–4963 (2019)

[44] Choquette-Choo, C.A., Tramer, F., Carlini, N., Papernot, N.: Label-only membership inference attacks. In: International Conference on Machine Learning, pp. 1964–1974 (2021). PMLR

[45] Li, Z., Zhang, Y.: Membership leakage in label-only exposures. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pp. 880–895 (2021)

[46] Kotyan, S., Vargas, D.V.: Adversarial robustness assessment. PloS one **17**(4), 0265723 (2022)

[47] Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv:1412.6572 (2014)

[48] Leurent, G., Peyrin, T.: SHA-1 is a shambles: First Chosen-Prefix collision on SHA-1 and application to the PGP web of trust. In: 29th USENIX Security Symposium (USENIX Security 20), pp. 1839–1856 (2020)

[49] Kamal, P.: A study on the security of password hashing based on gpu based, password cracking using high-performance cloud computing. Master's thesis, St. Cloud State. St. Cloud, Minnesota, USA. (2017)

[50] Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., Roli, F.: Evasion attacks against machine learning at test time. In: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13, pp. 387–402 (2013). Springer

[51] Bradburn, M.J., Clark, T.G., Love, S.B., Altman, D.G.: Survival analysis part

II: multivariate data analysis–an introduction to concepts and methods. British journal of cancer **89**(3), 431–436 (2003)

[52] Kleinbaum, D.G., Klein, M.: Survival Analysis a Self-learning Text. Springer, New York, NY, USA (2012)

[53] Tan, J., Yang, J., Wu, S., Chen, G., Zhao, J.: A critical look at the current train/test split in machine learning. arXiv preprint arXiv:2106.04525 (2021)

[54] Dohmatob, E.: Generalized No Free Lunch Theorem for Adversarial Robustness. In: Proceedings of the 36th International Conference on Machine Learning. PMLR, vol. 97 (2019)

[55] Meyers, C., Reza, M., Löfstedt, T., Elmroth, E.: A systematic approach to robustness modelling. In: Accepted for International Conference on Machine Learning and Cybernetics (2023)

[56] Ozaki, Y., Tanigaki, Y., Watanabe, S., Onishi, M.: Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference, pp. 533–541 (2020)

[57] Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2623–2631 (2019)

[58] Watanabe, S.: Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. arXiv preprint arXiv:2304.11127 (2023)

[59] Austin, P.C., Harrell Jr, F.E., Klaveren, D.: Graphical calibration curves and the integrated calibration index (ICI) for survival models. Statistics in Medicine **39**(21), 2714–2742 (2020)

[60] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

[61] Amaral, M., Chen, H., Chiba, T., Nakazawa, R., Choochotkaew, S., Lee, E.K., Eilam, T.: Kepler: A framework to calculate the energy consumption of containerized applications. In: 2023 IEEE 16th International Conference on Cloud Computing (CLOUD), pp. 69–71 (2023)

[62] Sedghpour, M.R.S., Townend, P.: Service mesh and ebpf-powered microservices: A survey and future directions. In: 2022 IEEE International Conference on Service-Oriented System Engineering (SOSE), pp. 176–184 (2022). https://doi.org/10.1109/SOSE55356.2022.00027

[63] Zitzler, E., Knowles, J., Thiele, L.: Quality assessment of pareto set approximations. Multiobjective optimization: Interactive and evolutionary approaches, 373–404 (2008)

[64] Legriel, J., Le Guernic, C., Cotton, S., Maler, O.: Approximating the pareto front of multi-criteria optimization problems. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, pp. 69–83 (2010). Springer

[65] Deng, L.: The MNIST database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine **29**(6), 141–142 (2012)

[66] Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Technical report, Toronto, ON, Canada (2009)

[67] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv:1706.06083 (2017)

[68] Moosavi-Dezfooli, S.-M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2574–2582 (2016)

[69] Xu, W., Evans, D., Qi, Y.: Feature squeezing: Detecting adversarial examples in deep neural networks. arXiv:1704.01155 (2017)

[70] Svedin, M., Chien, S.W., Chikafa, G., Jansson, N., Podobas, A.: Benchmarking the nvidia gpu lineage: From early k80 to modern a100 with asynchronous memory transfers. In: Proceedings of the 11th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies, pp. 1–6 (2021)

[71] Xu, R., Han, F., Ta, Q.: Deep learning at scale on nvidia v100 accelerators. In: 2018 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS), pp. 23–32 (2018). IEEE

[72] Google: GPU pricing. Compute Engine: Virtual Machines (VMS). google cloud. Google. https://cloud.google.com/compute/gpus-pricing

[73] Google: GPU pricing. Compute Engine: Virtual Machines (VMS). google cloud. Google. https://cloud.google.com/compute/vm-instance-pricing#accelerator-optimised

[74] Davidson-Pilon, C.: lifelines: survival analysis in python. Journal of Open Source Software **4**(40), 1317 (2019) https://doi.org/10.21105/joss.01317

[75] Dayarathna, M., Wen, Y., Fan, R.: Data center energy consumption modeling: A survey. IEEE Communications surveys & tutorials **18**(1), 732–794 (2015)

[76] Schmoor, C., Sauerbrei, W., Schumacher, M.: Sample size considerations for the

evaluation of prognostic factors in survival analysis. Statistics in medicine **19**(4), 441–452 (2000)

[77] Lachin, J.M.: Introduction to sample size determination and power analysis for clinical trials. Controlled clinical trials **2**(2), 93–113 (1981)

Paper V

A Tiny, Client-Side Classifier.

Charles Meyers, Aaron P. MacSween, Erik Elmroth, and Tommy Löfstedt.

Manuscript, Umeå University, Sweden, 2024.

Is it possible to build a classifier that circumvents the inherent weaknesses of centralised training paradigms?

A Tiny, Client-Side Classifier

C. Meyers, A. P. MacSween, E. Elmroth, and T. Löfstedt

May 8, 2025

Abstract

The recent developments in machine learning have highlighted a conflict between online platforms and their users in terms of privacy. The importance of user privacy and the struggle for power over user data has been intensified as regulators and operators attempt to police the online platforms. As users have become increasingly aware of privacy issues, client-side data storage, management, and analysis have become a favoured approach to large-scale centralised machine learning. However, state-of-the-art machine learning methods require vast amounts of labelled user data, making them unsuitable for models that reside client-side and only have access to a single user's data. State-of-the-art methods are also computationally expensive, which degrades the user experience on compute-limited hardware and also reduces battery life. A recent alternative approach has proven remarkably successful in classification tasks across a wide variety of data—using a compression-based distance measure (called normalized compression distance) to measure the distance between generic objects in classical distance-based machine learning methods. In this work, we demonstrate that the normalized compression distance is actually not a metric; develop it for the wider context of kernel methods to allow modelling of complex data; and present techniques to improve the training time of models that use this distance measure. We show that the normalised compression distance works as well as and sometimes better than other metrics and kernels—without incurring additional computational costs and in spite of the lack of formal metric properties. The end results is a simple model with remarkable accuracy even when trained on a very small number of samples allowing for models that are small and effective enough to run entirely on a client device using only user-supplied data.

1 Introduction

Modern machine learning (ML) methods have demonstrated remarkable efficacy across many domains. However, they often have large numbers of parameters, and thus also require large numbers of samples to train on [1]. This aggregation of vast amounts of data creates numerous privacy, safety, and security threats [2] that are consequences of large-scale user data collection, for instance by online platform operators (see Section 1.1 for more details). We evaluate and extend prior work on compression-based distance measures by incorporating them into novel kernel methods, enabling efficient classification even with limited training samples. When using small training sets and small and efficient models, they can be trained entirely on a client device without sharing private user data with anyone—allowing the model builder to circumvent many weaknesses associated with state-of-the-art methods [2, 3, 1]. We demonstrate the efficacy of the proposed approach in the context of malware detection, network intrusion detection, and spam detection.

1.1 Threat Model

In the context of online platforms, data are collected from end-users using dubious amounts of consent [4] and aggregated at massive scales [1]. Such data collection on online platforms often creates privacy, safety, and security risks [5, 6]. One such privacy risk is the periodic attempts by regulators to weaken encryption standards [7] and create backdoors to user devices. Platform operators and governments discuss best-practices for scanning client devices for illegal or "offensive" content [2, 8], but existing proposals are no less risky to user privacy, safety, or security than weakening encryption. Privacy experts have denounced such approaches for numerous reasons: the ease of *extracting* private training data [9, 10, 11] or the model itself [10, 12, 13, 14, 15], the ability of a malicious user to induce false positives for other users when the model is trained on user data [*poisoning* attacks; 16, 17, 18, 19, 20, 21, 22, 23, 24], and the triviality of simply *evading* the detection mechanism [25, 26, 27, 28, 6, 5, 29, 27].

Current platform solutions often rely on large-scale ML methods that are trained on vast amounts of user data and federated across devices [8]—an approach that has been criticized by privacy experts [2]. Examples of security risks include attacks against ML systems that target a model during train-

ing [30], prediction [28, 29, 25], and deployment [31, 32]. Even when access to a model by an adversary is limited, it is possible to induce a misclassification [27], reverse engineer the model [33], determine the model weights [13], or infer the class-membership of new samples [34]. This raises profound questions for safety-critical systems [6] and legal questions about access and control of the underlying data [35, 36]. Additionally, it has been shown that finding prototypical meta samples from the training set of large-scale ML models is trivial [37, 6]. Even if the attacker only has access to a typical application programming interface (API), there are reliable ways to fool the model [27].

Many privacy experts warn of the potential for any hypothetical *centralized* content-censoring system not only because of the potential failures due to adversaries mentioned above, but also because of the potential these systems to be used for mass surveillance and censorship [2]. In short, distributed and centralised training paradigms are both inherently fragile to malicious users and dangerous for user privacy, even if the resulting model lives on the user device (rather than in the cloud).

1.2 Motivations

In contrast to many state-of-the-art methods, this work proposes a lightweight, client-side approach to content filtering that does not rely on large-scale data collection.

Recently, Jiang *et al.* [38] proposed a remarkably successful approach to "parameter free" classification, dubbed NCD-KNN, that exploits a compression-based distance measure, the *normalized compression distance* [NCD; 39], to classify objects using the k-nearest neighbours (KNN) method [40]. NCD-KNN is known to work well even when trained on small numbers of samples [41]. By building a model that is accurate on a small number of samples, we can fulfil the goal of training a ML model entirely on a client device using data generated by a single user. An additional goal of this work was to evaluate the efficacy of NCD in general and to extend it to kernel methods in particular.

While other research examined topics like image classification [42], molecular property classification [43], and text classification [44], the ability of NCD to classify datasets that contain strings, numeric values, and categorical data (heterogenous datasets) has remained unexplored.

Additionally, the original NCD work [39] included an error term that

is usually ignored in recent research [42, 43, 44, 38]. To the best of our knowledge, no research has addressed the problem of negative values for NCD. The original authors [39] asserted that NCD is always positive when using perfect compressors. However, this is clearly demonstrated to be false in Lemma 1 when using imperfect compressors (which is what we will have in practice) .

Prior works about NCD have primarily focused on distance-based ML methods (*e.g.*, KNN), which limits both the class of methods that can be considered and the types of data that can be effectively analysed. A key motivation for this work was to extend the use of NCD beyond distance-based methods by developing a novel kernel-based formulation. This significantly broadens the applicability of NCD, making it suitable for a wider range of machine learning techniques where traditional distance-based approaches cannot be used.

1.3 Contributions

To use NCD, the model builder must choose a compression algorithm. While the effect of various compressors has been explored, in part, before [45], we expand the analysis by Cebrián *et al.* [45] to more recent compression algorithms and also offer additional run-time improvements over previous implementations [38].

The NCD-KNN method has shown very strong performance across several benchmarks, but prior implementations [38] are not appropriate for real-time settings due to unnecessary repeated computations. We therefore propose several run-time improvements and modifications, outlined in Section 3.1.

Further, we show that NCD is not a metric [42, 43, 44, 38], which means that applying ML methods blindly can lead to erroneous results (*e.g.*, by incorrectly ordering the nearest neighbours). In this work we demonstrate this non-metric behaviour in Lemma 1 and propose techniques to mitigate the effects of this behaviour in Section 3.1, effectively making the modified NCD "more like a metric".

Additionally, we expand the notion of NCD [42, 43, 44, 39, 38, 38] to kernels (Section 3.2) thus allowing for this method to be used with other models besides KNN. We thus extend NCD to reproducing kernel Hilbert spaces and hence more elaborate ML methods—allowing its use in a broader set of ML methods and to model more complex decision boundaries.

Hence, in this paper, we:

- Demonstrate that the normalized compression distance is not a metric (Lemma 1) and propose techniques to make it behave "more like a metric" (Section 3.1.2).
- Propose the use of NCD as a kernel and evaluate its efficacy.
- Develop classifiers (evaluations with KNN, logistic regression, and support vector machines (SVC)) that offers large run-time improvements over a reference implementation [38](Section 3.1).
- Evaluate and empirically show the efficacy of the proposed NCD-based classifiers across multiple binary classification tasks (Section 4).

2 Background

In the sections below, we describe and define the NCD, outline the NCD-KNN method proposed by Jiang *et al.* [38], outline several other string metrics, and discuss the distance matrix and how to efficiently compute it,

2.1 Normalized Compression Distance

NCD has been demonstrated to be a *universal* measure of similarity between two objects [39]—where a value of 0 denotes equivalence and a value of 1 denotes complete dissimilarity. The NCD is defined as [39]

$$\mathrm{NCD}(x, x') = \frac{|\mathcal{C}(xx')| - \min\{|\mathcal{C}(x)|, |\mathcal{C}(x')|\}}{\max\{|\mathcal{C}(x)|, |\mathcal{C}(x')|\}} + \varepsilon, \tag{1}$$

where $|\mathcal{C}(z)|$ is the length of the compressed form of the data, z, using compression algorithm, $\mathcal{C}$, the notation xx' denotes the concatenation of strings x and x', and $\varepsilon \geq 0$ is an error term accounting for imperfect compression algorithms [39]. The error term is usually assumed to be small relative to the other term.

NCD requires a choice of compression algorithm, and we evaluated the `gzip` [46], `bz2` [47], and `brotli` [48] compressors. To distinguish between the use of these different compressors, a subscript is used such that $\mathrm{NCD}_{\mathtt{gzip}}$ denotes the use of the `gzip` compressor.

While NCD is often discussed as a measure of distance [42, 43, 44, 38, 39], it does in fact not adhere to the axioms of metric spaces. A function, d :

$X \times X \to \mathbb{R}$, associated with a set of points, X, where $\mathbb{R}$ denotes the set of real numbers, is said to be a metric if the following four axioms hold for all $x_1, x_2, x_3 \in X$ [49]:

$$\begin{aligned} \text{Zero Axiom:} \quad & d(x_1, x_2) = 0 \iff x_1 = x_2 && (2) \\ \text{Non-negativity Axiom:} \quad & d(x_1, x_2) \geq 0 && (3) \\ \text{Symmetry Axiom:} \quad & d(x_1, x_2) = d(x_2, x_1) && (4) \\ \text{Triangle Inequality:} \quad & d(x_1, x_3) \leq d(x_1, x_2) + d(x_2, x_3). && (5) \end{aligned}$$

Much of the literature devoted to NCD has treated it as a proper metric [42, 43, 44, 38]. However, as we show now, it is not.

Lemma 1. *When using* `gzip`*,* `bz2`*, and* `brotli` *compressors, NCD does not adhere to the axioms in Equations 2–5, and is thus not a metric.*

Proof. We show in what follows that NCD fails to adhere to the axioms for metrics by counter-examples.

Zero axiom: The following counter-examples violate the zero axiom:

$$\text{NCD}_{\text{gzip}}(A, A) = 0.05, \quad \text{NCD}_{\text{bz2}}(B, G) = 0, \quad \text{and} \quad \text{NCD}_{\text{brotli}}(X, X) = 0.2.$$

Non-negativity axiom: The following counter-examples violate the non-negativity axiom:

$$\text{NCD}_{\text{gzip}}(AAAA, AAAA) = -0.04,$$

$$\text{NCD}_{\text{bz2}}(AABABAA, BAABAAB) = -0.03,$$

and

$$\text{NCD}_{\text{brotli}}(CCCCBBCCC, CBCCCBBCCC) = -0.08.$$

Symmetry axiom: The following counter-examples violate the symmetry axiom:

$$\text{NCD}_{\text{gzip}}(AA, BAA) = 0.13 \neq \text{NCD}_{\text{gzip}}(BAA, AA) = 0.04,$$

$$\text{NCD}_{\text{bz2}}(AA, AAB) = 0.11 \neq \text{NCD}_{\text{bz2}}(AAB, AA) = 0.00,$$

and

$$\text{NCD}_{\text{brotli}}(AAAAAAA, B) = 0.6 \neq \text{NCD}_{\text{brotli}}(B, AAAAAAA) = 0.7.$$

Triangle Inequality: The following counter-examples violate the triangle inequality:

$$\mathrm{NCD}_{\mathrm{gzip}}(AAA, A) > \mathrm{NCD}_{\mathrm{gzip}}(AAA, AAAA) + \mathrm{NCD}_{\mathrm{gzip}}(AAAA, A),$$

$$\mathrm{NCD}_{\mathrm{bz2}}(BC, AN) > \mathrm{NCD}_{\mathrm{bz2}}(BC, J) + \mathrm{NCD}_{\mathrm{bz2}}(J, AN),$$

and if

$$x_1 = CAAAACAA, \quad x_2 = CAC, \quad \text{and} \quad x_3 = CCACCCACCC,$$

then

$$\mathrm{NCD}_{\mathrm{bz2}}(x_1, x_3) > \mathrm{NCD}_{\mathrm{bz2}}(x_1, x_2) + \mathrm{NCD}_{\mathrm{bz2}}(x_2, x_3).$$

□

2.2 Other String Metrics

To model datasets that comprise strings, several existing measures of distance between strings are routinely used [50]. To evaluate the relative performance of the NCD metric, we compared it to several other common measures of string distance:

- *Levenshtein* is the "edit distance" or minimum number of single-character edits to transform one string into another [51].
- *Lev Ratio* is the Levenshtein distance divided by the total length of the longer string [50].
- *Hamming* is the number of character positions where two strings differ.
- *Ham Ratio* is the number of character positions where two strings differ, divided by the length of the longer string.

2.3 Calculating the distance matrix

In what follows, the pairwise distances between two sets of samples (denoted X and X') is collected in a a *distance matrix*, D. When computing the value of $\mathrm{NCD}(x, x')$, it is necessary to calculate the values of $\mathcal{C}(x)$ and $\mathcal{C}(x')$. To classify a sample, Jiang *et al.* [38] iterated over all elements in the sets X and X', where X would be a set of samples with known labels and X' one

with unknown labels, such that for each $x_i \in X$ the compression $\mathcal{C}(x_i)$ was computed repeatedly for each $x'_j \in X'$. Because computational time scales linearly with the size of both X and X', the run time is thus $\mathcal{O}(|X| \cdot |X'|)$ for both compute and memory, where $|X|$ is the cardinality of the set X. Clearly, if the number of samples in X and X' are large, then run-time becomes a concern. We propose some modifications to how the pairwise distances are computed and which pairwise distances are computed to reduce the computational costs and also make NCD behave "more like a metric", which is outlined in Section 3.1.

3 Methods

This section outlines proposed modifications to NCD and Jiang's NCD-KNN method [38] before outlining the data and experiments used to verify the efficacy of said modifications.

3.1 Proposed Modifications to NCD

Jiang *et al.*'s implementation of the NCD-KNN method [38] becomes inefficient because it includes many repeated computations. To minimize run-time, we first propose a simple improvement in Section 3.1.1. Second, we propose several modifications that symmetrise the distance matrix, intended to ensure adherence to the symmetry axiom (Equation 4). Then, in an effort to improve the adherence to the zero axiom (Equation 2) we propose a modification for the special case of $x = x'$ in Section 3.1.3. Finally, we outline the proposed way to use NCD as a kernel in Section 3.2.

3.1.1 Pre-computing the Compression vector

When computing the value of $\text{NCD}(x, x')$, it is necessary to calculate the values of $\mathcal{C}(x)$ and $\mathcal{C}(x')$ as in Equation 1. For each $x \in X$, the $\mathcal{C}(x)$ would be computed repeatedly when computing the pairwise NCD distances between $x \in X$ and the elements in X'. Instead of recomputing the $\mathcal{C}(x)$ repeatedly, it is much more efficient to pre-compute the compressed versions of each element in X and X' only once. Since compressing the input samples is the most costly part of this computation, this saves substantial run-time as demonstrated in Section 4.

3.1.2 Symmetrisation

We propose three new ways to induce "more" adherence to the axioms in Equations 2–5 and compare their efficacy and run-time in Section 4.

For the purposes of the experiments below, the method proposed by Jiang *et al.* [38] is denoted "Vanilla" and computes the pairwise distances between two sets by naively computing every element of a distance matrix using the NCD function in Equation 1.

The second method, proposed here as a modification to NCD, assumes symmetry by only computing the lower triangular part of the distance matrix and then reflecting those values about the diagonal, instead of computing the entire distance matrix; this method is denoted "Assumed." Hence, in a distance matrix, D, the "Assumed" method computes the lower triangular part of D and then let

$$D_{i,j} = D_{j,i}, \tag{6}$$

which effectively halves the computational cost of computing the distance matrix.

In the third method, proposed here, symmetry is *enforced* by sorting the inputs of NCD alphanumerically before computing the distance between them. This approach thus ensures symmetry during prediction as well as training. This method is denoted "Enforced", and also effectively halves the computational cost of computing the distance matrix since again $D_{i,j} = D_{j,i}$.

The fourth method, also proposed here, computes the *average* value of $\mathrm{NCD}(x, x')$ and $\mathrm{NCD}(x', x)$. The average of $\mathrm{NCD}(x, x')$ and $\mathrm{NCD}(x', x)$ is

$$\overline{\mathrm{NCD}}(x, x') = \frac{\mathrm{NCD}(x, x') + \mathrm{NCD}(x', x)}{2}, \tag{7}$$

which can be simplified to

$$\overline{\mathrm{NCD}}(x, y) = \frac{\frac{\mathcal{C}(xx')+\mathcal{C}(x'x)}{2} - \min[\mathcal{C}(x), \mathcal{C}(x')]}{\max[\mathcal{C}(x), \mathcal{C}(x')]} + \varepsilon, \tag{8}$$

which clearly leads to $D_{i,j} = D_{j,i}$. The simplification in Equation 8 thus only includes one additional compression, increasing the computational cost by roughly 20% instead of doubling the computational cost as in Equation 7. This method is denoted "Averaged".

3.1.3 Zero-axiom check

A simple means to ensure that the output of NCD is zero when the two inputs are equal, we propose to check for this case before any distances are computed and return zero if the inputs are equal. We denote this modification the "Zero-axiom check". The Zero-axiom check was performed with the "Assumed" and "Enforced" methods, but not the "Averaged" method, assuming that the error associated with calculating $\mathrm{NCD}(x, x')$ would cancel out the error associated with $\mathrm{NCD}(x, x')$.

3.2 Kernelisation

We propose to use NCD to construct an approximate kernel, allowing NCD to be used with a much larger set of ML methods than as a distance. For this purpose, a kernel is defined as a function, $k : X \times X \to \mathbb{R}$, such that

$$k(x, x') := \langle \phi(x), \phi(x') \rangle \tag{9}$$

for all $x, x' \in X$, where $\phi : X \to Y$ is a feature function (a function extracting features from its inputs), and $\langle \cdot, \cdot \rangle$ denotes an inner product in the feature space, Y. The i-th row and j-th column of a kernel matrix, K, is

$$K_{ij} = k(x_i, x_j).$$

The radial basis function (RBF) kernel [40], also known as the Gaussian kernel (when the distance is the Euclidean distance) is defined as

$$k(x, x') = \exp\left(-\frac{d(x, x')^2}{\lambda}\right), \tag{10}$$

where λ is a tunable parameter (denoted a *length scale*) that controls how quickly the kernel function decreases as a function of the distance between points, *i.e.*, determines the influence of individual points on neighbouring points. We thus propose to use NCD as the distance function, d, in the kernel in Equation 10. The RBF kernel is particularly effective as it is known to be a universal function approximator [52].

The Hamming kernel [53], based on the Hamming distance between two strings or binary vectors, is defined as

$$k(x, x') = 1 - \lambda \frac{d_H(x, x')}{\max(|x|, |x'|)}, \tag{11}$$

where $d_H(x, x')$ denotes the Hamming distance, $\max(|x|, |x'|)$ denotes the length of the longer string, and λ is a tunable parameter that controls the sensitivity of the kernel to differences between input vectors. Smaller values of λ cause the kernel to decay more rapidly as the number of differing positions increases, thereby emphasizing exact or near-exact matches. We propose to use this kernel in settings where inputs are strings or binary representations, as it naturally captures similarity through positional agreement. Like the RBF kernel, the Hamming kernel belongs to the class of positive-definite kernels and has been shown to be effective at classification tasks [54].

Euclidean distances can be computed in the feature space by using a kernel. We see that

$$\begin{aligned} d(x, x') &= \|\phi(x) - \phi(x')\|_2^2 \\ &= \langle \phi(x) - \phi(x'), \phi(x) - \phi(x') \rangle \\ &= \langle \phi(x), \phi(x) \rangle + \langle \phi(x'), \phi(x') \rangle - 2\langle \phi(x), \phi(x') \rangle \\ &= k(x, x) + k(x', x') - 2k(x, x'), \end{aligned}$$

and denote this distance as the *kernel distance*. For the RBF kernel, when the symmetry and the zero axioms hold, we have that $k(x, x) = k(x', x') = 1$, and the kernel distance can be computed efficiently as

$$d_k(x, x') = 2 - 2k(x, x'). \tag{12}$$

This formulation was used to extend NCD-KNN to kernels, *i.e.*, the kernel distance in Equation 12 was used together with the RBF kernel in Equation 10, as well as with the Hamming kernel in Equation 11. Logistic regression and SVCs were trained on the kernel matrices formed from Equations 10 & 11.

3.3 Data

Several open datasets were used to evaluate the efficacy of NCD in the context of heterogeneous tabular and text data.

We used the KDD-NSL data, which is a log of system process data for both regular users (denoted benign) and malicious software (denoted adversarial) [55]. It includes 6072 samples and 41 features that encapsulate the behaviour of both benign software and malware. KDD-NSL includes software

protocol, system error rate, whether the process has root privileges, and the number of files accessed by the process.

We also used the DDoS IoT dataset [56], which includes information collected from network packet headers of adversarial and benign users across many types of DDoS attacks. Specific features include source IP address, source port, destination IP address, destination port, and network protocol among a total of 90 features across more than 40 million samples, collected from both benign users and malicious traffic.

We used the Truthseeker dataset [57], which includes 134 thousand messages from Twitter users with a label provided by the data distributors, and a label that encodes whether or not a given user was a suspected bot.

Finally, we used the SMS Spam dataset [58] which includes SMS messages and a label indicating whether or not a message is spam across 5575 samples.

For several of the datasets, malicious examples were rare compared to the number of benign examples. To address the class imbalances, each dataset was under-sampled [59] using the `imblearn` package [60] to reduce bias towards the majority class and to ensure metrics like accuracy are meaningful. For each dataset, model, symmetrisation method, and distance metric, 1000 samples from each dataset were used to conduct five-fold cross validation, yielding five disjoint validation sets of 200 samples each. Accuracy, distance matrix computation time, model training times, and prediction times were recorded for each of the five cross-validation folds. In an effort to represent both text and numerical data as strings, the rows of each numerical dataset were extracted as lists in `Python` and then those lists were cast directly to strings for the DDoS, KDD-NSL, and SMS Spam datasets.

3.4 Experiments

We evaluated the proposed methodology using the described datasets, models, symmetrisation methods ("Vanilla", "Assumed", "Enforced", and "Averaged"), and metrics (NCD_{gzip}, NCD_{bz2}, and NCD_{brotli}, Levenshtein distance, a normalised Levenshtein distance (labelled Lev Ratio), Hamming distance, and a normalised Hamming distance (labelled Ham Ratio)). After generating the 5-fold cross validation sets for each dataset, the distance matrices for each symmetrisation method and metric were computed, as outlined in Sections 2.3 and 3.1.2. Additionally, kernel matrices were computed as described in Section 3.2. The classifiers used were KNN, logistic regression, and SVC, as implemented in `scikit-learn` [61]. For KNN, we both used NCD

directly and used the kernel distance from Equation 12 computed using both the RBF and Hamming kernels. For (kernel) logistic regression and SVC, the RBF kernel in Equation 10 and the Hamming kernel in Equation 11 were used with the NCD distance.

Each model was tuned using the hyper-parameters outlined in the following. For NCD, all metrics (NCD, Hamming distance, Levenshtein distance, etc) were used to compute kernel matrices as per Equations 10 and 12. In addition, the Hamming kernel from Equation 11 was used as a baseline [53]. Both kernels have a hyper-parameter, λ, that must be tuned; λ was was evaluated in powers of 10 in the range $[10^{-3}, 10^{3}]$.

KNN requires the model builder to specify the number of nearest neighbours. In our experiments, $k \in \{1, 3, 5, 7, 11\}$, as odd numbers means there were no ties (for our binary classification task). In logistic regression, an ℓ_2 penalty was used as well as a configuration without any penalty. The coefficient of the penalty was set to powers of 10 in the range $[10^{-3}, 10^{3}]$. The `SAGA` solver [62] was used for logistic regression, with a tolerance of 10^{-4}. The penalty term in the SVC was varied in the range $[10^{-3}, 10^{3}]$ for each power of ten.

To find the most appropriate set of hyper-parameters, each of the dataset-model-symmetrisation-metric combinations enumerated above were evaluated using a grid search and 5-fold cross-validation. Then, the best-fit model was chosen for each dataset-model-symmetrisation-metric configuration by finding the configuration(s) with the highest mean cross-validation accuracy and choosing the model with the smallest standard deviation in the case ties. After fitting each model to each dataset, symmetrisation method, measure of distance, and model-dependent hyper-parameters, the best-fit models were repeatedly trained on $m \in \{10, 20, 35, 60, 100, 200, 500, 1000\}$ samples and evaluated against the 200 withheld samples during cross-validation to evaluate the ability of the model to generalise even when trained on a small number of samples.

In addition to the classification experiments above, the distance matrices calculated from those datasets were exhaustively checked for adherence to the axioms in Equations 2–4 and the percentage of violations for each axiom was recorded. In addition, 100k randomly selected 3-tuples from each distance matrix were sampled to compute the percentage of samples that violate Equation 5. An additional synthetic dataset comprised of 100k 3-tuples was generated from short, alphabetic strings using the uppercase English alphabet and a maximum string size of 144 characters (the length of a "tweet") for

each metric and symmetrisation method and used to calculate the probability of axiom violations.

4 Results and Discussion

In this section, the results from the aforementioned experiments are discussed.

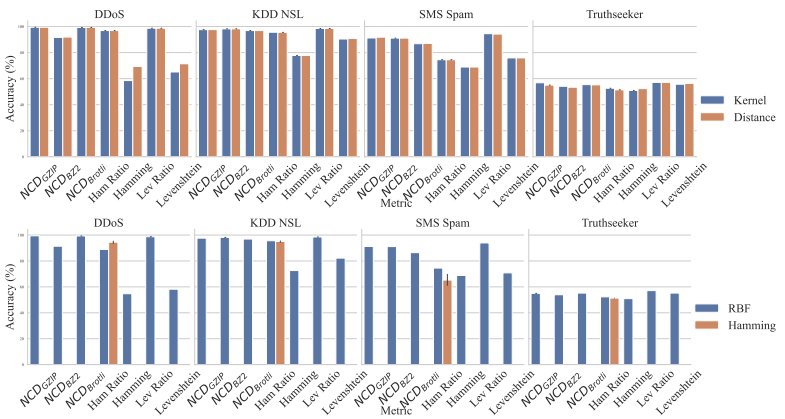


Figure 1: The 5-fold accuracy across each dataset (columns), distance metric (first-axis), whether or not the distance or kernel matrix was used (top plot, colour), and symmetrisation method (bottom plot, colour). For both plots, the bars represent the mean accuracy and the error bars represent 95% confidence intervals.

Figure 1 depicts the 5-five fold accuracies of the best-fit model for each dataset-metric-model-kernel combination. The error bars reflect the 95% confidence interval of the mean accuracy, computed across five cross-validated folds. The top of Figure 1 compares accuracy of the proposed kernelised KNN to the distance-based KNN proposed by Jiang *et al.* [38] for a variety of string metrics (first axis). The bottom of Figure 1 compares the RBF kernel (Equation 10) to the Hamming kernel (Equation 11) for all three kernelised ML models. It is clear that the kernel method is consistent with the distance method (top of Figure 1), apart from the un-normalised Levenshtein and

Hamming distances, wherein the distance method is superior for the DDoS dataset. Additionally, the RBF kernel has accuracy that is consistent with the Hamming kernel (bottom of Figure 1) and at least one version of NCD outperforms the Hamming ratio in every dataset. Therefore, it is clear that the proposed kernelised classification model is as accurate at the distance-based KNN [38] (top of Figure 1) and can also outperform the Hamming kernel (bottom of Figure 1).

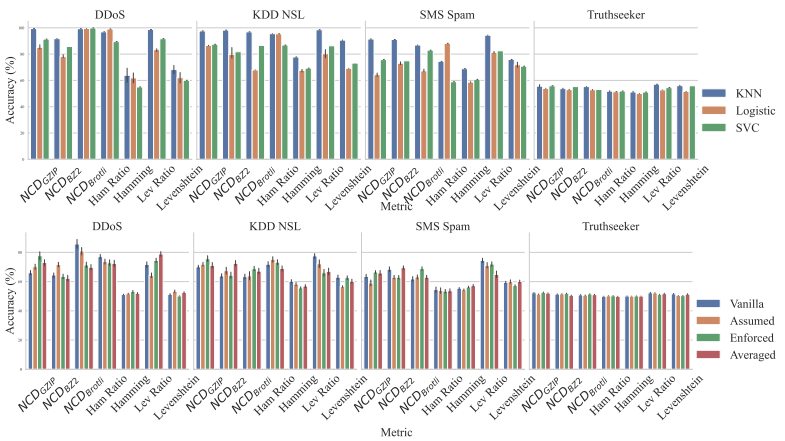


Figure 2: The accuracy across each dataset (columns), distance metric (first-axis), model (top plot, colour), and symmetrisation method (bottom plot, colour). For both plots, the bars represent the mean accuracy and the error bars represent 95% confidence intervals. Only KNN was used in the top plot as it is the only tested model that is meant to use distance matrices and the results depict the 5-fold accuracies for the best-fit model of each model-dataset-metric combination before (orange) and after kernelisation (blue).

Figure 2 shows the classifier performance across each dataset and string metric (including NCD, using various compressors) for both the kernel and distance matrices (top row), all models (middle row) and all symmetrisation methods (bottom row). The top sub-figure groups the results by whether or not the distance or kernel matrix was used, the middle sub-figure groups the results by model type using colour, and the bottom figure groups the results by symmetrisation method. From these results it is clear that KNN

is the most consistently accurate (middle of Figure 2) for NCD and that the distance matrix method tends to out-perform the kernel method (top of Figure 2).

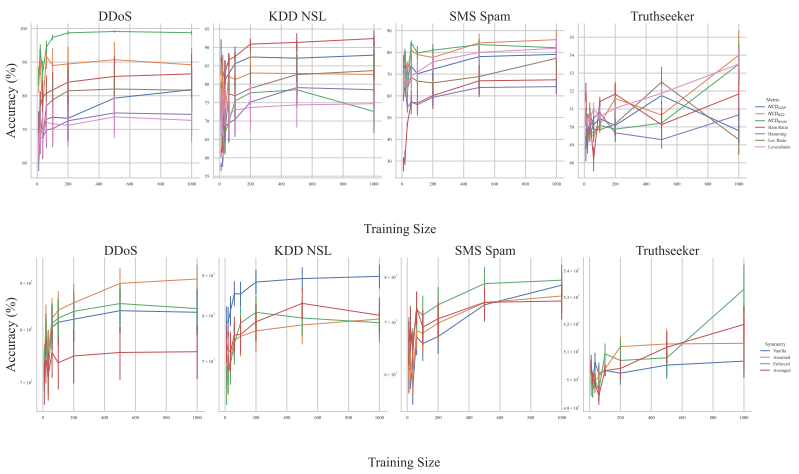


Figure 3: Accuracy across varying training sample sizes, computed on 200 samples withheld during cross-validation using the best-fit model for each model–metric–symmetrisation configuration. *Top:* Accuracy is grouped by evaluation metric (colour), showing how different metrics behave across datasets (columns) and training sizes, averaged over models and symmetrisation methods. *Bottom:* Accuracy is grouped by symmetrisation method (colour), illustrating how different symmetrisation choices impact performance across datasets, averaged over models and metrics. Each line represents the mean accuracy, and error bars indicate the 95% confidence interval of the mean.

Figure 3 depicts the accuracy of the best-fit models as the number of training samples is varied (first-axis). The top of Figure 3 groups the results by metric and the bottom groups the results by symmetrisation method, using colour to differentiate the groups. For the DDoS dataset, it is clear from the top subplot of Figure 3 that NCD improves the accuracy when a small number of samples are used to train the model compared to other string metrics. However, results are more mixed for other datasets. Additionally,

we can see that non-Vanilla symmetrisation methods tend to outperform the Vanilla variety on DDoS, SMS-Spam, and Truthseeker, although this is not significant.

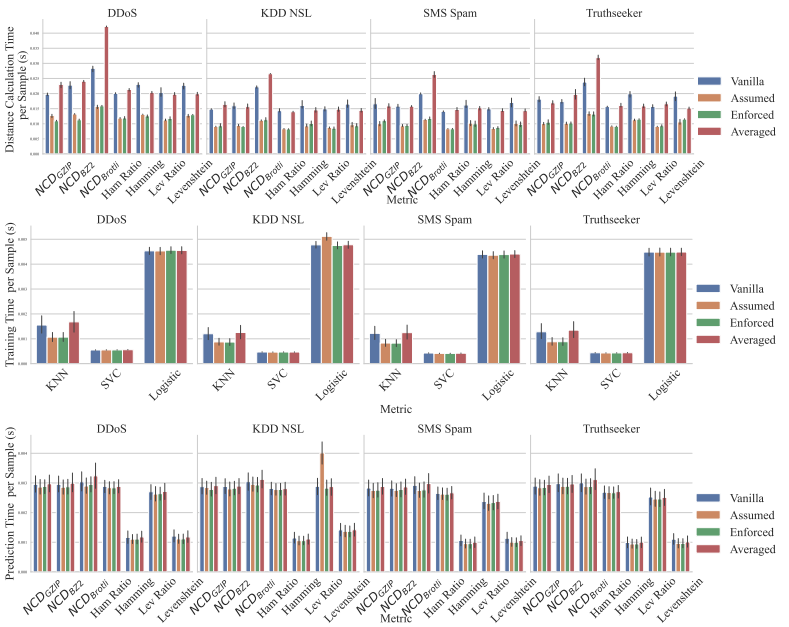


Figure 4: Performance times for various stages of the model evaluation process, computed across different metrics, datasets, and methods. *Top:* Distance matrix calculation time per sample averaged over all dataset-method-metric combinations. *Middle:* Training time per sample, measured after computing the distance matrix, averaged over all dataset-method-metric-model combinations. *Bottom:* Prediction time per sample, also after distance matrix computation, showing the time required for predictions averaged over all dataset-method-metric-model combinations. For all three sub-plots, the colour reflects the symmetrisation method. The firs-axis shows the metric in the top and bottom plots, but the middle plot depicts the training time as it varies more by model than metric. The top of each bar reflects the mean of the measured time and the error bars are 95% confidence intervals.

Figure 4 shows the run-times associated with calculating the distance between two samples (top), training the model on a single sample (middle), or inferring the classification of a new sample (bottom). The importance of the symmetrisation methods is to reduce the run-time. It is clear from Figure 4 that choice of symmetrisation method has a substantial effect on run-time, with the "Assumed" and "Enforced" symmetrisation methods taking roughly half as long per sample as the "Vanilla" version and the "Averaged" version taking slightly longer. During training (assuming the distance or kernel-matrix is pre-computed), the variance between models is far large than the variance induced by the symmetrisation methods (see middle row of Figure 4). During prediction, the differences between symmetrisation methods are much less pronounced, owing to the smaller number of samples—instead, the variance in prediction times is predominated by the run-time of the distance function as the variance within a given metric is much smaller than the variance between them (see the bottom row of Figure 4).

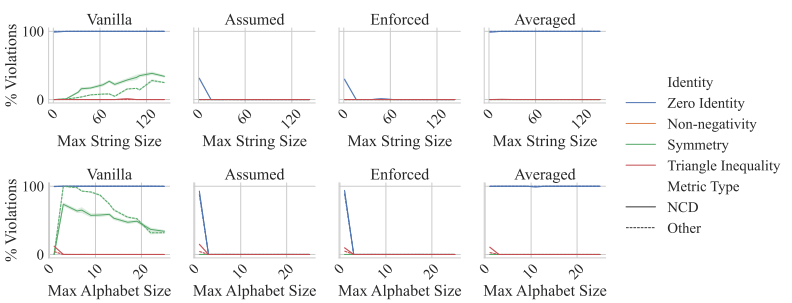


Figure 5: Percentage of violations of Axioms 2–5 among 100,000 random strings (from the standard English alphabet), using four symmetrisation methods: vanilla (top row), assumed symmetry, enforced symmetry, and averaging (bottom row). The top plots vary max string size; bottom plots vary alphabet size. Default settings: max string size is 144, alphabet size is 26. Colours indicate axioms; solid lines indicate NCD (Equation 1) and dashed lines are for other string metrics (Section 2.2).

Figures 5–6 display how the symmetrisation methods influence to what degree a given metric adheres to the axioms outlined in Equations 2–5 and show that they enforce adherence to the symmetric and zero-axiom as ex-

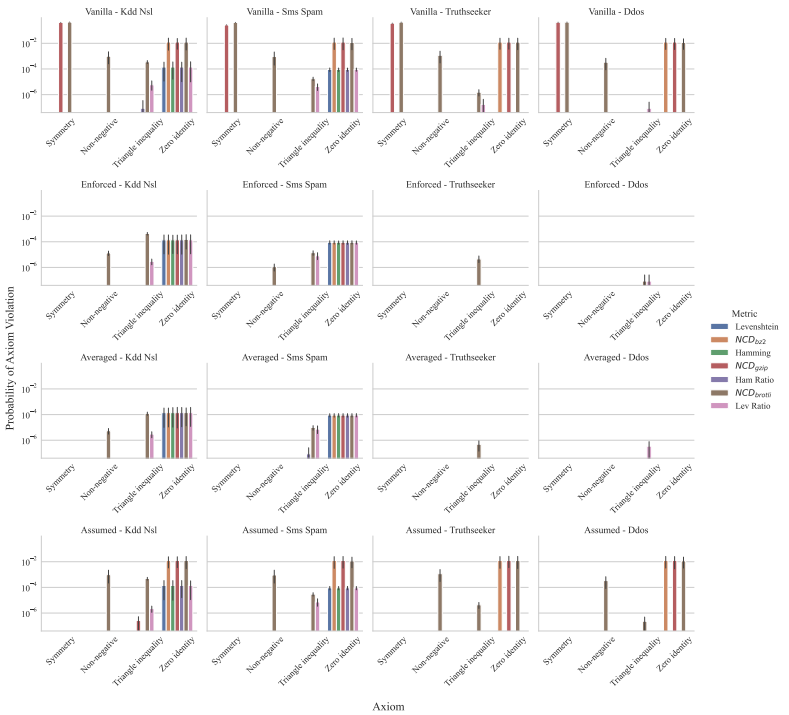


Figure 6: Percentage of examples found that violate the assumptions outlined in Equations 2–5 using the vanilla (top row), assumed symmetry (second row), enforced (third row), and averaged (bottom row) methods on the training matrices for each of the outlined datasets. Each column is dedicated to a dataset and each model is given a column. The first axis displays which of the axioms is violated and the colour indicates which symmetrisation method was used. Since evaluating all possible distance 3-tuples would be computationally infeasible for even hundreds of samples, three samples were sampled 100 thousand times such that no 3-tuple was repeated.

pected. It is clear from these figures that the averaging, enforcing, or assuming symmetry works quite well, thus fulfilling Equation 4. Likewise, Equation 2 acts as expected (compare "Vanilla" and "Averaged" with the other two methods). This is true on randomly constructed strings (Figure 5) as well as strings collected from the aforementioned datasets (Figure 6). It is clear that this does not correspond to a decrease in accuracy (see Figure 2) and that these modifications can significantly improve run-time (see Figure 4 in Figures 5–6).

5 Limitations

While the methods presented in this work demonstrate strong performance across various tasks and datasets, several limitations should be acknowledged.

To further improve run-time performance, compression algorithms optimised for graphics processing units (GPUs) have been developed [63]. These are likely to outperform the CPU-based implementations used in this paper when applied to large-scale datasets. Incorporating such GPU-accelerated compressors could significantly reduce processing time and improve scalability.

The model tuning parameters used in this study were limited in scope. While the selected configurations were sufficient to demonstrate key insights, a broader hyperparameter search may reveal additional trade-offs or improvements in accuracy and efficiency.

This work focused on a specific set of compression algorithms. Although other compressors exist–including those optimised for specific data types [63, 64, 65]—-they were considered out of scope for this study. Future work could explore the impact of alternative compressors on both accuracy and run-time. Likewise, the data preprocessing step used for heterogeneous tabular data was intentionally kept simple—each row was cast to a Python list then cast again as a string. While effective for our purposes (see Figure 1), this approach is crude and may obscure structural or semantic relationships within the data by including extraneous characters like commas and brackets (used to delineate and denote a list in Python, respectively). As always, the best implementation will be determined by the data and its schema. More sophisticated encoding methods—such as schema-aware parsing or embedding techniques—could improve performance, particularly on more complex or high-dimensional datasets.

6 Conclusion

Overall, we see that NCD is at least as accurate as other string metrics (top of Figure 2), despite not being a true metric (Lemma 1). Furthermore, we see that the proposed symmetrisation methods are quite effective—sometimes even outperforming the "Vanilla" method (bottom of Figure 2). Kernelised NCD is effective even when only a small number of samples are used to train the model, making it ideal for lightweight, client-side deployments (top of Figure 3) and our extensions (Section 3.1) do not significantly change this result (bottom of Figure 3). NCD can reach more than 95% accuracy on even tens of samples (Figure 3). It is clear from Figure 4 that the "Assumed" and "Enforced" symmetrisation methods proposed in this work are superior to those found in the literature by decreasing the run-time without penalising accuracy (bottom of Figure 2). In some cases, the "Averaged" symmetrisation method offers superior accuracy over the other methods (Figure 2, bottom) while inducing a marginal cost of only a few milliseconds (Figure 4, top). In other cases, "Enforced" and "Assumed" offer both superior accuracies (Figure 2, bottom) and run-times (Figure 4, top and bottom).

The proposed model is a real-time, client-side classification method that can be trained on a very small number of samples—potentially collected from only a single user. This reduces the attack surface to only adversaries that have access to data that users generally consider private (*e.g.*, the contents of a message). That is, the attack surface is reduced and can be unique to each user, session, or device by using data from each user in isolation and training a model on that user's device. Client-side models would only need to share a single bit (the classification label in a binary classification context) with the platform operator—allowing those operators to serve user-specific contents without exfiltrating private data from their entire user base. However, not training a model on large amounts of collected user data means that such hypothetical client-side methods are at risk of not performing at the level of state-of-the-art large-scale methods. The end result is a model with a substantially reduced attack surface that is nevertheless accurate, but also simple, generally applicable, and very fast.

References

[1] R. Desislavov, F. Martínez-Plumed, J. Hernández-Orallo, Compute and energy consumption trends in deep learning inference, arXiv preprint arXiv:2109.05472 (2021).

[2] H. Abelson, R. Anderson, S. M. Bellovin, J. Benaloh, M. Blaze, J. Callas, W. Diffie, S. Landau, P. G. Neumann, R. L. Rivest, J. I. Schiller, B. Schneier, V. Teague, C. Troncoso, Bugs in our pockets: the risks of client-side scanning, Journal of Cybersecurity 10 (1) (Jan. 2024).

[3] Goldman Sachs Research, AI is poised to drive 160% increase in data center power demand, Goldman Sachs (May 2024).
URL `https://www.goldmansachs.com/insights/articles/AI-poised-to-drive-160-increase-in-power-demand`

[4] M. Nouwens, I. Liccardi, M. Veale, D. Karger, L. Kagal, Dark patterns after the GDPR: Scraping consent pop-ups and demonstrating their influence, in: Proceedings of the 2020 CHI conference on human factors in computing systems, 2020.

[5] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, D. Mukhopadhyay, Adversarial attacks and defences: A survey, arXiv:1810.00069 [cs, stat] (2018).

[6] C. Meyers, T. Löfstedt, E. Elmroth, Safety-critical computer vision: An empirical survey of adversarial evasion attacks and defenses on computer vision systems, Artificial Intelligence Review (2023).

[7] Amnesty International, Encryption—a matter of human rights (2016).
URL `https://www.amnesty.nl/content/uploads/2016/03/160322_encryption_-_a_matter_of_human_rights_-_def.pdf?x12112`

[8] Apple Inc., CSAM detection: A technical summary (2021).
URL `https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf`

[9] C. A. Choquette-Choo, F. Tramer, N. Carlini, N. Papernot, Label-only membership inference attacks, in: International conference on machine learning, PMLR, 2021, pp. 1964–1974.

[10] M. Fredrikson, S. Jha, T. Ristenpart, Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15, ACM Press, Denver, Colorado, USA, 2015.

[11] Z. Li, Y. Zhang, Membership leakage in label-only exposures, in: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, 2021, pp. 880–895.

[12] T. Orekondy, B. Schiele, M. Fritz, Knockoff nets: Stealing functionality of black-box models, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 4954–4963.

[13] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, N. Papernot, High accuracy and high fidelity extraction of neural networks, in: 29th USENIX security symposium (USENIX Security 20), 2020, pp. 1345–1362.

[14] J. R. Correia-Silva, R. F. Berriel, C. Badue, A. F. De Souza, T. Oliveira-Santos, Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data, in: 2018 International joint conference on neural networks (IJCNN), IEEE, 2018, pp. 1–8.

[15] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership inference attacks against machine learning models, in: 2017 IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 3–18.

[16] A. Rawat, K. Levacher, M. Sinn, The devil is in the gan: backdoor attacks and defenses in deep generative models, in: European Symposium on Research in Computer Security, Springer, 2022, pp. 776–783.

[17] R. Shokri, et al., Bypassing backdoor detection algorithms in deep learning, in: 2020 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE, 2020, pp. 175–183.

[18] T. Gu, B. Dolan-Gavitt, S. Garg, Badnets: Identifying vulnerabilities in the machine learning model supply chain, arXiv preprint arXiv:1708.06733 (2017).

[19] A. Saha, A. Subramanya, H. Pirsiavash, Hidden trigger backdoor attacks, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 34, 2020, pp. 11957–11965.

[20] H. Aghakhani, D. Meng, Y.-X. Wang, C. Kruegel, G. Vigna, Bullseye polytope: A scalable clean-label poisoning attack with improved transferability, in: 2021 IEEE European symposium on security and privacy (EuroS&P), IEEE, 2021, pp. 159–178.

[21] A. Turner, D. Tsipras, A. Madry, Clean-label backdoor attacks (2018).

[22] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, T. Goldstein, Poison frogs! targeted clean-label poisoning attacks on neural networks, Advances in neural information processing systems 31 (2018).

[23] J. Geiping, L. Fowl, W. R. Huang, W. Czaja, G. Taylor, M. Moeller, T. Goldstein, Witches' brew: Industrial scale data poisoning via gradient matching, arXiv preprint arXiv:2009.02276 (2020).

[24] H. Souri, L. Fowl, R. Chellappa, M. Goldblum, T. Goldstein, Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch, Advances in Neural Information Processing Systems 35 (2022) 19165–19178.

[25] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: 2017 IEEE symposium on security and privacy (sp), Ieee, 2017, pp. 39–57.

[26] E. Dohmatob, Generalized No Free Lunch Theorem for Adversarial Robustness, in: Proceedings of the 36th International Conference on Machine Learning, Vol. 97 of PMLR, 2019.

[27] J. Chen, M. I. Jordan, M. J. Wainwright, HopSkipJumpAttack: A query-efficient decision-based attack, in: IEEE symposium on security and privacy (sp), IEEE, 2020, pp. 1277–1294.

[28] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, F. Roli, Evasion attacks against machine learning at test time, 2013, pp. 387–402.

[29] S.-M. Moosavi-Dezfooli, A. Fawzi, P. Frossard, Deepfool: a simple and accurate method to fool deep neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2574–2582.

[30] B. Biggio, B. Nelson, P. Laskov, Poisoning Attacks against Support Vector Machines, 2013.

[31] A. Aljuhani, Machine learning approaches for combating distributed denial of service attacks in modern networking environments, IEEE Access 9 (2021).

[32] P. M. Santos, B. Manoj, M. Sadeghi, E. G. Larsson, Universal adversarial attacks on neural networks for power allocation in a massive mimo system, IEEE Wireless Communications Letters 11 (1) (2021) 67–71.

[33] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, N. Papernot, High accuracy and high fidelity extraction of neural networks, in: 29th USENIX security symposium (USENIX Security 20), 2020, pp. 1345–1362.

[34] J. W. Bentley, D. Gibney, G. Hoppenworth, S. K. Jha, Quantifying membership inference vulnerability via generalization gap and other model metrics, arXiv preprint arXiv:2009.05669 (2020).

[35] L. Mitrou, Data protection, artificial intelligence and cognitive services: is the general data protection regulation (gdpr)'artificial intelligence-proof'?, Artificial Intelligence and Cognitive Services: Is the General Data Protection Regulation (GDPR)'Artificial Intelligence-Proof (2018).

[36] M. Marks, C. E. Haupt, Ai chatbots, health privacy, and challenges to hipaa compliance, Jama (2023).

[37] C. Meyers, M. R. S. Sedghpour, T. Lofstedt, E. Elmroth, A training rate and survival heuristic for inference and robustness evaluation (trashfire), in: 2024 International Conference on Machine Learning and Cybernetics (ICMLC), 2024, pp. 613–623.

[38] Z. Jiang, M. Y. Yang, M. Tsirlin, R. Tang, J. Lin, Less is more: Parameter-free text classification with gzip, arXiv preprint arXiv:2212.09410 (2022).

[39] M. Li, X. Chen, X. Li, B. Ma, P. Vitanyi, The similarity metric, IEEE Transactions on Information Theory 50 (12) (2004).

[40] S. Shalev-Shwartz, S. Ben-David, Understanding machine learning: From theory to algorithms, Cambridge university press, Cambridge, UK., 2014.

[41] M. Scilipoti, M. Fuster, R. Ramele, A strong inductive bias: Gzip for binary image classification, arXiv preprint arXiv:2401.07392 (2024).

[42] J. Opitz, Gzip versus bag-of-words for text classification with knn, arXiv preprint arXiv:2307.15002 (2023).

[43] J. Weinreich, D. Probst, Parameter-free molecular classification and regression with gzip (2023).

[44] K. Nishida, R. Banno, K. Fujimura, T. Hoshide, Tweet classification by data compression, in: Proceedings of the 2011 international workshop on DETecting and Exploiting Cultural diversiTy on the social web, 2011, pp. 29–34.

[45] M. Cebrián, M. Alfonseca, A. Ortega, Common pitfalls using the normalized compression distance: What to watch out for in a compressor (2005).

[46] Free Software Foundation, GZIP: Gnu zip, https://www.gnu.org/software/gzip/manual/gzip.html (2009–2023).

[47] muraroa.demon.co.uk, bz2 (Jul. 1998).
URL `https://web.archive.org/web/19980704181204/http://www.muraroa.demon.co.uk/`

[48] Google Inc., Github - google/brotli: Brotli compression format.
URL `https://github.com/google/brotli`

[49] D. Burago, A course in metric geometry, American Mathematical Society (2001).

[50] rapidfuzz, Levenshtein, https://rapidfuzzrap.github.io/Levenshtein/ (2021).

[51] G. Navarro, A guided tour to approximate string matching, ACM computing surveys (CSUR) 33 (1) (2001) 31–88.

[52] B. Hammer, K. Gersmann, A note on the universal approximation capability of support vector machines, neural processing letters 17 (2003) 43–53.

[53] K. T. Phelps, M. LeVan, Kernels of nonlinear hamming codes, Designs, Codes and Cryptography 6 (3) (1995) 247–257.

[54] M. Norouzi, D. J. Fleet, R. R. Salakhutdinov, Hamming distance metric learning, Advances in neural information processing systems 25 (2012).

[55] G. Mohi-ud din, NSL-KDD, IEEE Dataport (2018).

[56] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, A. A. Ghorbani, Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment, Sensors 23 (13) (2023) 5941.

[57] M. Khalil, M. Azzeh, Fake news detection models using the largest social media ground-truth dataset (truthseeker), International Journal of Speech Technology (2024) 1–16.

[58] T. Almeida, J. Hidalgo, SMS Spam Collection, UCI Machine Learning Repository (2011).

[59] S. Yen, Y. Lee, Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset, Lecture notes in control and information sciences 344 (2006) 731.

[60] G. Lemaître, F. Nogueira, C. K. Aridas, Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning, Journal of Machine Learning Research 18 (17) (2017).

[61] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[62] A. Defazio, F. Bach, S. Lacoste-Julien, Saga: A fast incremental gradient method with support for non-strongly convex composite objectives, Advances in neural information processing systems 27 (2014).

[63] R. A. Patel, Y. Zhang, J. Mak, A. Davidson, J. D. Owens, Parallel lossless data compression on the GPU, IEEE, 2012.

[64] K. Brandenburg, Mp3 and aac explained, in: Audio Engineering Society Conference: 17th International Conference: High-Quality Audio Coding, Audio Engineering Society, 1999.

[65] V. Sze, M. Budagavi, G. J. Sullivan, High efficiency video coding (HEVC), Integrated circuit and systems, algorithms and architectures 39 (2014) 40.

Paper VI

Deckard: A tool for robust, declarative, and reproducible AI.

Charles Meyers.

Manuscript, Umeå University, Sweden, 2025.

How can we build auditable and reproducible tests to meet regulatory standards for safety critical systems?

deckard: A Declarative Tool for Machine Learning Robustness Evaluations

15 April 2025

Summary

The software package presented, called `deckard`, is a modular software toolkit designed to streamline and standardize experimentation in machine learning (ML) with a particular focuse on the adversarial scenario. It provides a flexible, extensible framework for defining, executing, and analyzing end-to-end ML pipelines in the context of a malicious actor. As it is built on top of the Hydra configuration system, deckard supports declarative YAML-based configuration of data preprocessing, model training, and adversarial attack pipelines, enabling reproducible, framework-agnostic experimentation across diverse ML settings.

In addition to configuration management, `deckard` includes a suite of utilities for distributed and parallel execution, automated hyperparameter optimisation, visualisation, and result aggregation. The tooling abstracts away much of the engineering overhead typically involved in adversarial ML research, allowing researchers to focus on algorithmic insights rather than implementation details. The presented software facilitates rigorous benchmarking by maintaining an auditable trace of configurations, random seeds, and intermediate outputs throughout the experimental lifecycle.

The system is compatible with a variety of ML frameworks and several classes of adversarial attacks, making it a suitable back-end for both large-scale automated testing and fine-grained empirical analysis. By providing a unified interface for experimental control, `deckard` accelerates the development and evaluation of robust models, and helps close the gap between research prototypes and verifiable, reproducible results.

Statement of need

While tools such as `mlflow` (Zaharia et al. 2018), Weights & Biases (Biewald 2020), `optuna` (Akiba et al. 2019), and Kubernetes (Kubernetes 2019) provide essential infrastructure for model tracking and experiment management, `deckard` occupies a different position in the ML ecosystem—focusing specifically on configurable, adversarially robust experimentation.

Unlike MLflow and Weights & Biases, which emphasize logging, visualization, and reproducibility for various ML frameworks, deckard enforces reproducibility by construction through its declarative, YAML-driven configuration system built on Facebook's hydra (Yadan 2019) configuration management tool. In contrast to cloud-management software like Kubernetes—which is a general-purpose container orchestration platform—`deckard` abstracts away orchestration details and offers native support for parallel and distributed experimentation, tailored to ML workflows involving attack/defense cycles, model retraining, or optimisation. While deckard integrates tightly with IBM's Adversarial Robustness Toolbox (Nicolae et al. 2018), the software is designed to be easily extensible to other attack frameworks. The human- and machine-readable parameter configuration system allows researchers to declaratively define end-to-end pipelines that span data sampling, preprocessing, model training, attack generation, defense evaluation, multi-objective optimisation, and visualisation. Tools like `ray` (Moritz et al. 2018), `optuna` (Akiba et al. 2019), or `nevergrad` (Bennet et al. 2021) offer components of this pipeline (*e.g.*, hyperparameter search or configuration management), but lack unified support for adversarial ML, verification, or auditability at scale. While `deckard` complements these existing tools, and in many cases can be integrated with them, its primary contribution is in automating and verifying adversarial ML experiments in a way that is both extensible and framework-agnostic.

Usage

Various versions of this software have been used in several recently published and not-yet-published works by the author of this paper, all of which are available in the `examples` folder in the source code repository https://github.com/simplymathematics/deckard. One published work, now reproducible via the `examples/attack_defence_survey` folder, includes a large survey of attacks and defences against canonical datasets and models (C. Meyers, Löfstedt, and Elmroth 2023). Another work analysed the run-time requirements of attacks against a particular model before and after retraining against those attacks (Charles Meyers and Elmroth 2024) (reproducible via `examples/retraining`). The next paper formalised a method for estimating the time-to-failure of a given model against a suite of attacks and introduce a metric that quantifies the ratio of attack and training cost (Meyers et al. 2023) (reproducible via `examples/survival_heuristic`). Furthermore, a not yet published work uses this time-to-failure model as a mechanism for analysing the cost efficacy of various hardware choices in the context of adversarial attacks (reproducible via `examples/power`) (C. Meyers et al. 2024). Another work exploits the tooling to train a custom model that is designed to run client-side by using compression algorithms to measure the distance between text (reproducible via `examples/compression`).

Experiment Management

Typically ML projects are composed of long and complex pipelines that are highly dependent on a number of parameters that must be configured by either the model builder or attacker. Due to the large scale and cost associated with training ML models, it is often necessary to tune a model using many individual model configurations (often called *hyper-parameters*). To determine adversarial robustness, one of many benchmark datasets is first sampled, then preprocessed, sent to a model, with optional pre- and post-processing defences, and then scored according to some chosen metric which may include the performance against any number of adversarial attacks. Each stage in this example pipeline might include tens or hundreds of possible sets of hyper-parameters that must be exhaustively tested. Furthermore, this problem scales drastically as we include more and more stages in a pipeline since each additional stage introduces a new combinatorial layer of complexity, rapidly expanding the total number of potential configurations that must be evaluated for robustness and be reproducible for posterity. Not only does `deckard` provide a standard way to document and configure these hyper-parameters, it gives each experiment an auditable identifier.

Reproducibility and Auditability

For ML, various regulatory and legal frameworks govern safety (The Parliament of the European Union 2024; "ISO 26262-1:2011, Road Vehicles — Functional Safety" 2018; *IEC 61508 Safety and Functional Safety* 2010; *IEC 62304 Medical Device Software - Software Life Cycle Processes* 2006), privacy (The Parliament of the European Union 2024; European Parliament and Council of the European Union 2016; Legislature of the United States 1996, 1998) and/or transparency (The Parliament of the European Union 2024; The Legislature of California 2024). The software package presented here provides a machine- and human-readable format for creating reproducible and auditable experiments as required by various regulations. In addition, several examples connected to both published and not-yet-published work live in the `examples` folder in the repository, allowing for easy reproducibility of several extensive sets of experiments across several popular ML software frameworks. The `power` example provides a reproducible way to run a suite of adversarial tests using popular cloud-based platforms and the `retraining` and `survival_heuristic` examples provide examples of both CPU and GPU-based parallelisation, respectively.

The `basics` subfolder provides a minimum working example for each of the supported ML frameworks: `tensorflow` (Abadi et al. 2015), `pytorch` (Paszke et al. 1912), `scikit-learn` (Pedregosa et al. 2011), and `keras` (Chollet 2015). The basics folder also provides examples of various classes of adversarial examples: *poisoning* attacks that change model behaviour by injecting data during training (Biggio, Nelson, and Laskov 2012), *inference* attacks (Li and Zhang 2021) that attempt to reverse engineer properties of the training data, *extraction* attacks that attempt to reverse engineer the model (Jagielski et al. 2020), and *evasion* attacks that

induce errors of classification during run-time (C. Meyers, Löfstedt, and Elmroth 2023). The parameters file for each experiment ensures that a given pipeline can be reproduced and the standardised format allows us to derive a hash value that is hard to forge but easy to verify. Not only does this hash serve as an identifier to track the state of an experiment, but also serves as a way to audit the parameters file for tampering. Likewise, by using `dvc` (DVC Authors 2023) to track any input or output files specified in the parameters file, the software associates each score file with a identifier that is easy to track and verify, but hard to forge—ensuring that forged or modified results are easy to spot in version-controlled experiment repository.

Parallel and Distributed Design

Since ML projects can exploit specialized hardware such as multi-core processors or GPUs, and often rely on clusters of machines for large-scale data processing, it was necessary to enable parallel and distributed experiment execution and model optimization. By leveraging the `hydra` configuration framework, `deckard` automatically supports optimization libraries like `nevergrad` (Bennet et al. 2021), `Adaptive Experimentation` (A. Developers 2025), and `optuna` (Akiba et al. 2019), making the software modular and extensible. Additionally, experiments can be managed using a variety of popular job schedulers, including `Ray` (Moritz et al. 2018), `Redis Queue` (Stamps 2025), and `slurm` (Yoo, Jette, and Grondona 2003) for distributd jobs or `joblib` (Joblib Developers 2025) for jobs on a single machine.

By using a declarative design, a given set of experiments can be specified once and executed seamlessly across different backends without modification to the underlying codebase. This makes `deckard` both adaptable and scalable, suitable for use on personal laptops, multi-node servers, or large-scale, high-performance clusters. When configured appropriately, experiment batches can be parallelized, enabling massive parameter sweeps, ensemble evaluations, or adversarial robustness tests to be executed in parallel—reducing turnaround time while maintaining strong guarantees on reproducibility and auditability. The design of the presented software prioritizes clarity and maintainability by capturing each experimental configuration as a YAML artifact, making both successful and failed runs equally traceable and shareable. This approach transforms experiment tracking from an afterthought into a first-class component of the trustworthy ML workflow.

Funding

Financial support has been provided in part by the Knut and Alice Wallenberg Foundation grant number 2019.0352 and by the eSSENCE Programme under the Swedish Government's Strategic Research Initiative.

Acknowledgements

The author would like to thank Aaron MacSween, Abel Souza, and Mohammad Saledghpour Reza for their guidance in software design principles. In particular, the author appreciates Mohammad's code and documentation regarding cloud-based and other Kubernetes deployments. The author would like to thanks his advisors, Erik Elmroth and Tommy Löfstedt for their research expertise, funding, and patience.

References

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, et al. 2015. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems."

Akiba, Takuya, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. "Optuna: A Next-Generation Hyperparameter Optimization Framework." *25th ACM SIGKDD*, 2623–31.

Bennet, Pauline, Carola Doerr, Antoine Moreau, Jeremy Rapin, Fabien Teytaud, and Olivier Teytaud. 2021. "Nevergrad: Black-Box Optimization Platform." *ACM SIGEVOlution* 14 (1): 8–15.

Biewald, Lukas. 2020. "Experiment Tracking with Weights and Biases." https://www.wandb.com/.

Biggio, Battista, Blaine Nelson, and Pavel Laskov. 2012. "Poisoning Attacks Against Support Vector Machines." *International Conference on Machine Learning*.

Charles Meyers, Tommy Löfstedt, and Erik Elmroth. 2024. "Massively Parallel Evasion Attacks and the Pitfalls of Adversarial Retraining." *EAI Endorsed Transactions on Internet of Things* 10.

Chollet, François. 2015. "Keras." *GitHub Repository*. https://github.com/fchollet/keras; GitHub.

Developers, Ax. 2025. "AX: Adaptive Experimentation Platform." https://ax.readthedocs.io/en/stable/#.

Developers, Joblib. 2025. "Joblib: Running Python Functions as Pipeline Jobs." https://joblib.readthedocs.io/en/stable/.

DVC Authors. 2023. "DVC–Data Version Control." Github. https://github.com/iterative/dvc.org.

European Parliament, and Council of the European Union. 2016. "Regulation (EU) 2016/679 of the European Parliament and of the Council." OJ L 119, 4.5.2016, p. 1–88. May 4, 2016. https://data.europa.eu/eli/reg/2016/679/oj.

IEC 61508 Safety and Functional Safety. 2010. 2nd ed. International Electrotechnical Commission.

IEC 62304 Medical Device Software - Software Life Cycle Processes. 2006. 2nd ed. International Electrotechnical Commission.

"ISO 26262-1:2011, Road Vehicles — Functional Safety." 2018. https://www.iso.org/standard/43464.html (visited 2022-04-20).

Jagielski, Matthew, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. 2020. "High Accuracy and High Fidelity Extraction of Neural Networks." In *29th USENIX Security Symposium (USENIX Security 20)*, 1345–62.

Kubernetes. 2019. "Kubernetes–an Open Source System for Managing Containerized Applications." Github. https://github.com/kubernetes/kubernetes.

Legislature of the United States. 1996. "Health Insurance Portability and Accountability Act."

Legislature of the United State. 1998. "Children's Online Privacy Protection Act."

Li, Zheng, and Yang Zhang. 2021. "Membership Leakage in Label-Only Exposures." In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 880–95.

Meyers, Charles, Tommy Löfstedt, and Erik Elmroth. 2023. "Safety-Critical Computer Vision: An Empirical Survey of Adversarial Evasion Attacks and Defenses on Computer Vision Systems." *Artificial Intelligence Review* 56 (Suppl 1): 217–51.

Meyers, Charles, Mohammad Reza Saleh Sedghpour, Tommy Löfstedt, and Erik Elmroth. 2024. "A Training Rate and Survival Heuristic for Inference and Robustness Evaluation (TRASHFIRE)." https://arxiv.org/abs/2401.13751.

Meyers, Reza, Löfstedt, and Elmroth. 2023. "A Systematic Approach to Robustness Modelling." *Springer Artificial Intelligence Review*.

Moritz, Philipp, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, et al. 2018. "Ray: A Distributed Framework for Emerging {AI} Applications." In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 561–77.

Nicolae, Maria-Irina, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, et al. 2018. "Adversarial Robustness Toolbox V1. 0.0." *arXiv Preprint arXiv:1807.01069*.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, et al. 1912. "Pytorch: An Imperative Style, High-Performance Deep Learning Library. arXiv 2019." *arXiv Preprint arXiv:1912.01703* 10.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12: 2825–30.

Stamps. 2025. "RQ: Easy Job Queues for Python." Stamps, an Indonesian CRM company. https://python-rq.org/.

The Legislature of California. 2024. "AB-2013 Generative Artificial Intelligence: Training Data Transparency."

The Parliament of the European Union. 2024. "High-Level Summary of the AI Act."

Yadan, Omry. 2019. "Hydra – a Framework for Elegantly Configuring Complex Applications." Github. https://github.com/facebookresearch/hydra.

Yoo, Andy B, Morris A Jette, and Mark Grondona. 2003. "Slurm: Simple Linux Utility for Resource Management." In *Workshop on Job Scheduling Strategies for Parallel Processing*, 44–60. Springer.

Zaharia, Matei, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, et al. 2018. "Accelerating the Machine Learning Lifecycle with MLflow." *IEEE Data Eng. Bull.* 41 (4): 39–45.