A TRAINING RATE AND SURVIVAL HEURISTIC FOR INFERENCE AND ROBUSTNESS EVALUATION (TRASHFIRE)

immediate

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 hyper-parameters 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 interdependent (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 50th 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 $\mathbb{E}[T]$. Counterintuitively, 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.

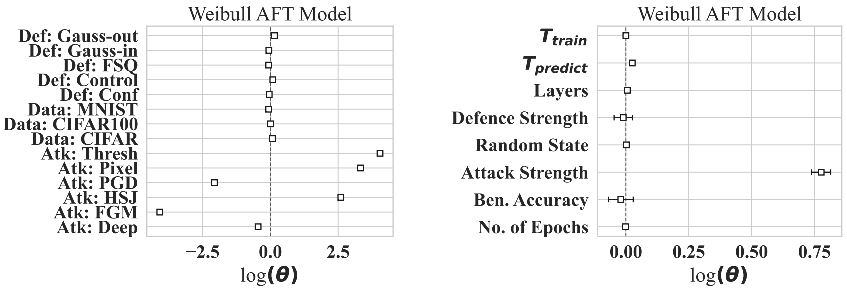

FIGURE 1. The coefficients represent the log scale effect of the dummy variables for dataset (Data), attack (Atk), and defence (Def) on the survival time, with a positive value indicating an increase in the survival time. The right plot depicts the covariates and the left plot depicts the dummy variables for the different attacks, defences, and datasets.

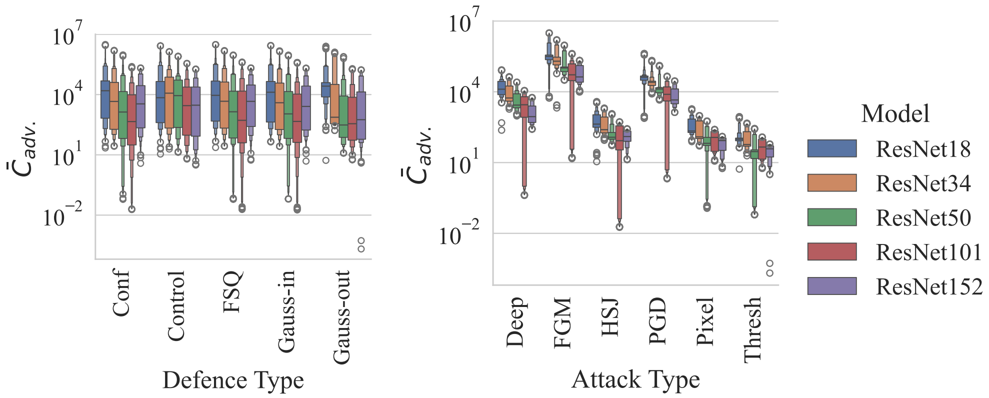

FIGURE 2. This figure depicts the TRASH metric that reflects the ratio of training-to-attack times, where a value ≫ 1 indicates an essential advantage for the attacker. The violin plots reflect the 95% confidence intervals for each tuned hyperparameter combination. Outliers are indicated with a circle.

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.