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/XX.X.X.XX

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].

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

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 hyperparameter sets, we propose a scalable framework for

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

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

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

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

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

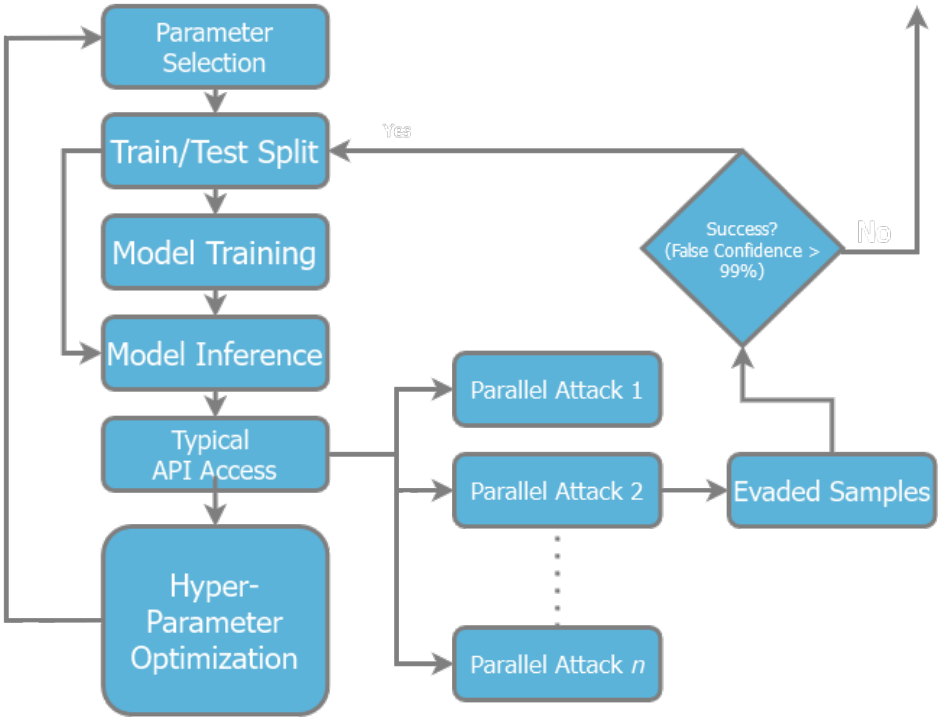


Figure 1. Massively Parallel Attack and Re-training Framework. This depicts our experimental pipeline wherein we build KSVMs with various kernels, run several attacks in parallel, and then evaluate the attacked samples on the models. Optionally, we collect highly confident false examples for adversarial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

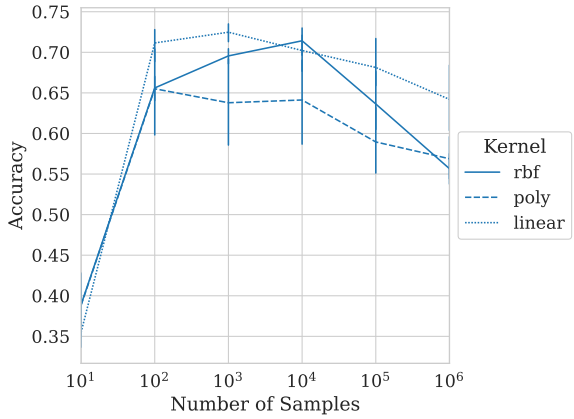
6.1. Performance:

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.

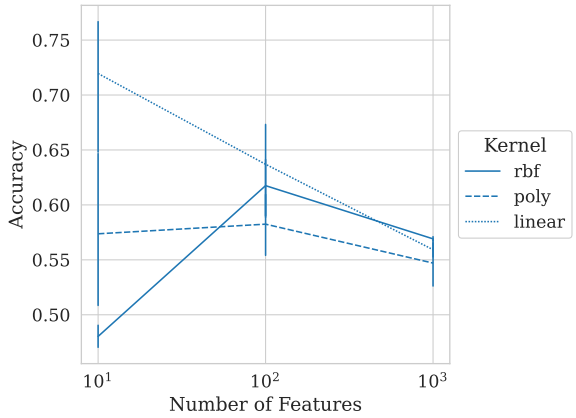
6.2. Accuracy and Robustness:

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

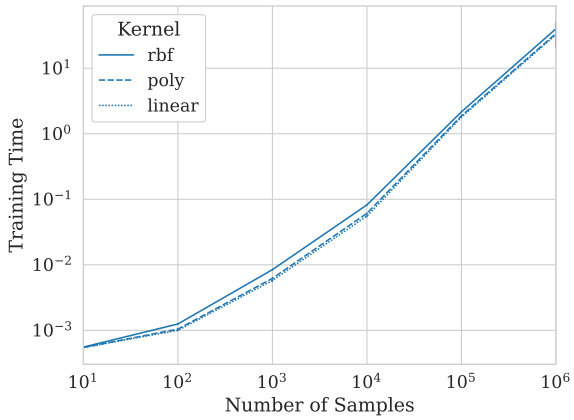
[2] https://joblib.readthedocs.io
[3] https://optuna.readthedocs.io
[4] https://hydra.cc
[5] https://dvc.org
[6] Our repository
[7] https://scikit-learn.org
[8] https://adversarial-robustness-toolbox.readthedocs.io
[9] Our Github Repository



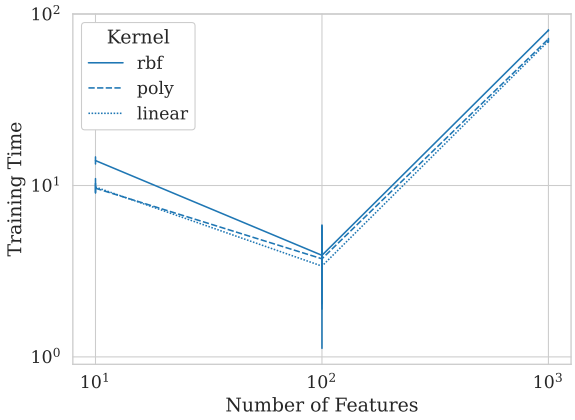
(a) Model Performance vs Database Size: This depicts the benign performance of a model (*e.g.*, accuracy on unperturbed data) when trained on databases of difference sizes for each tested kernel.



(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

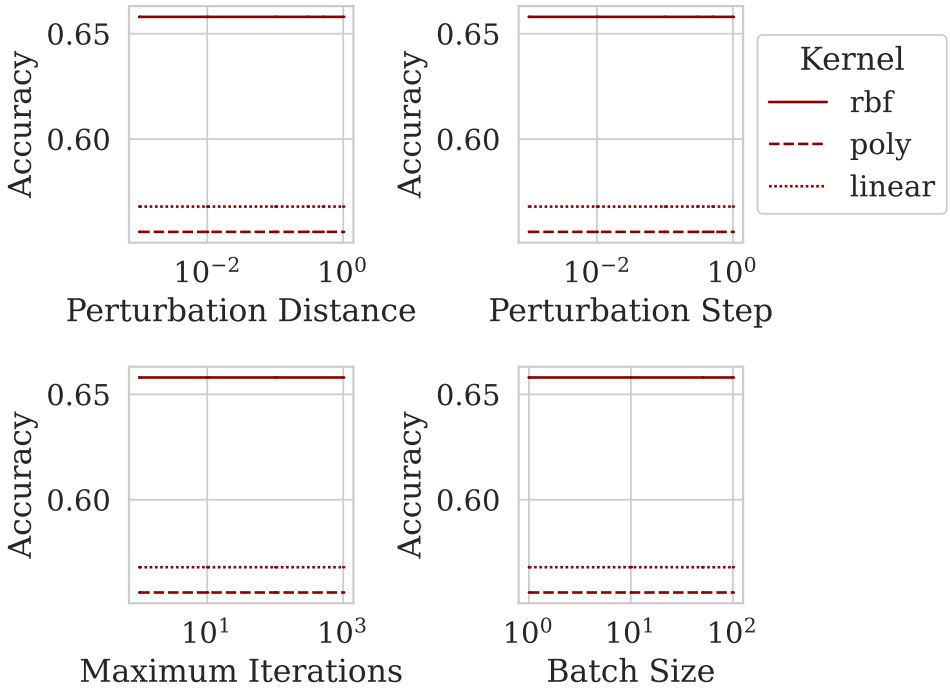


Figure 3. Attack Parameters vs Accuracy: This depicts how various attack hyperparameters change the accuracy.

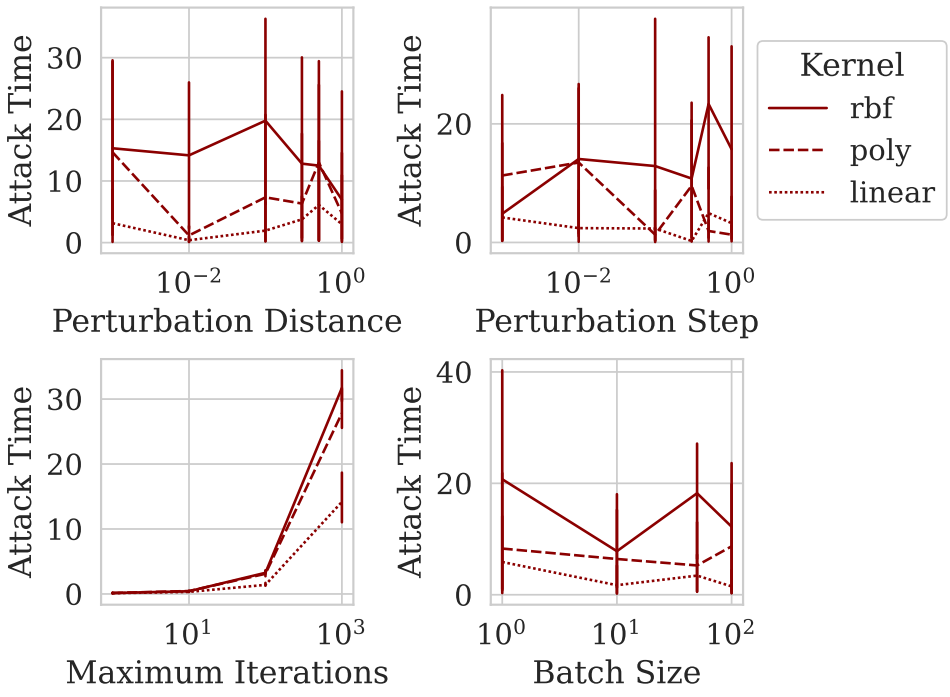


Figure 4. Attack Parameters vs Time: This depicts how various attack hyperparametrs change the run-time. The bars reflect the 95% confidence interval for all tested configurations.

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:

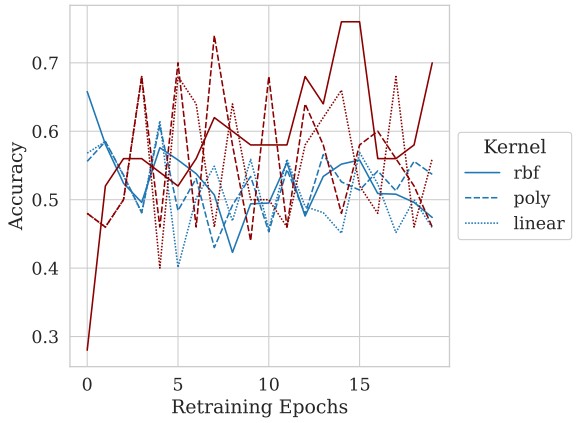
To reliably evaluate a model, we must look at how it performs across many attacks, so we measured both error (Figure 3) and false confidence (Figure 5c) for the entire attack space. We see that increasing either step size or perturbations tends to increase loss with a minimal perturbation size required 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.

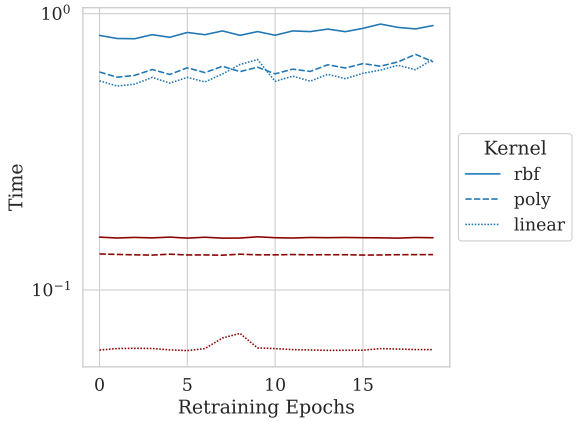
6.5. Adversarial Retraining

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.

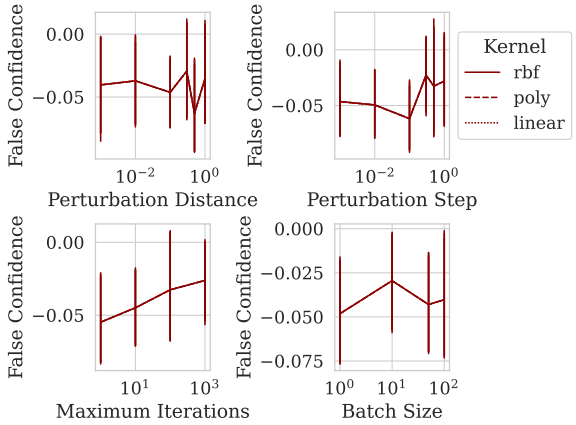
Furthermore, theoretical analysis shows that strong attacks will only cost more than model building for very large numbers of adversarial examples, which we've shown to be unnecessary when controlling for batch size. Since every model query has the potential to expose the attacker, a small number of 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



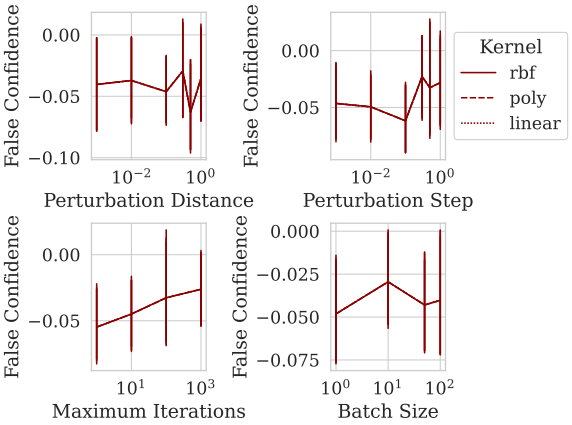
(a) Adversarial Retraining: Accuracy over several retraining cycles. The blue indicates the performance on unpertured data and the red indicates performance on the adversarial data.



(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 data-set. 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**.

Appendix A. KDD-NSL Dataset

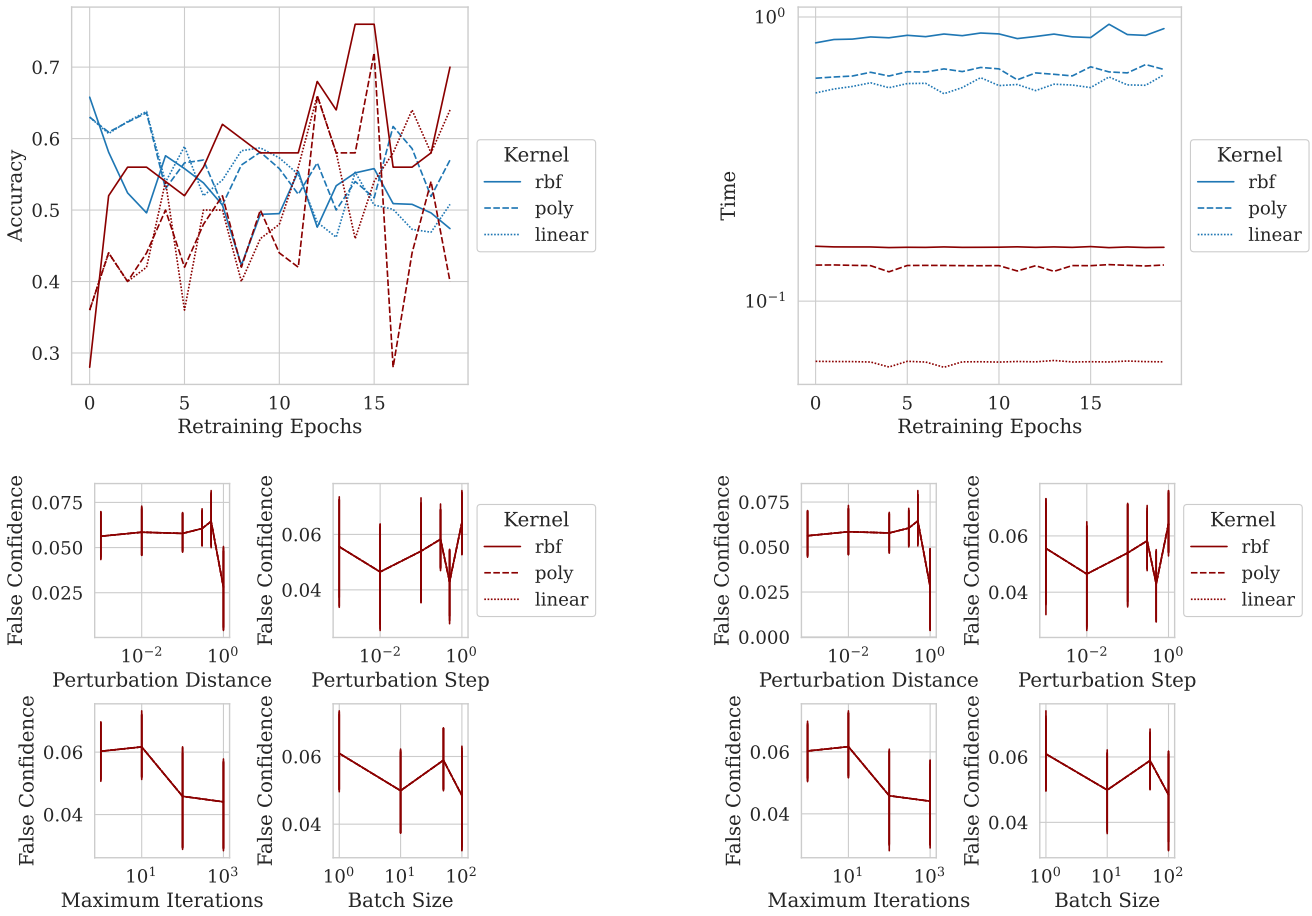


Figure A.1. Efficacy of Adversarial Retraining on KDD-NSL Dataset. The top left depicts the adversarial and benign accuracy over a number of retraining epochs. The top right depicts the per epoch training time as the number of training epochs increases. The bottom row depicts the false confidence before retraining (left) on strong adversarial examples and after (right). The bars reflect the 95% confidence interval for all tested configurations.

Appendix B. Truthseeker Dataset

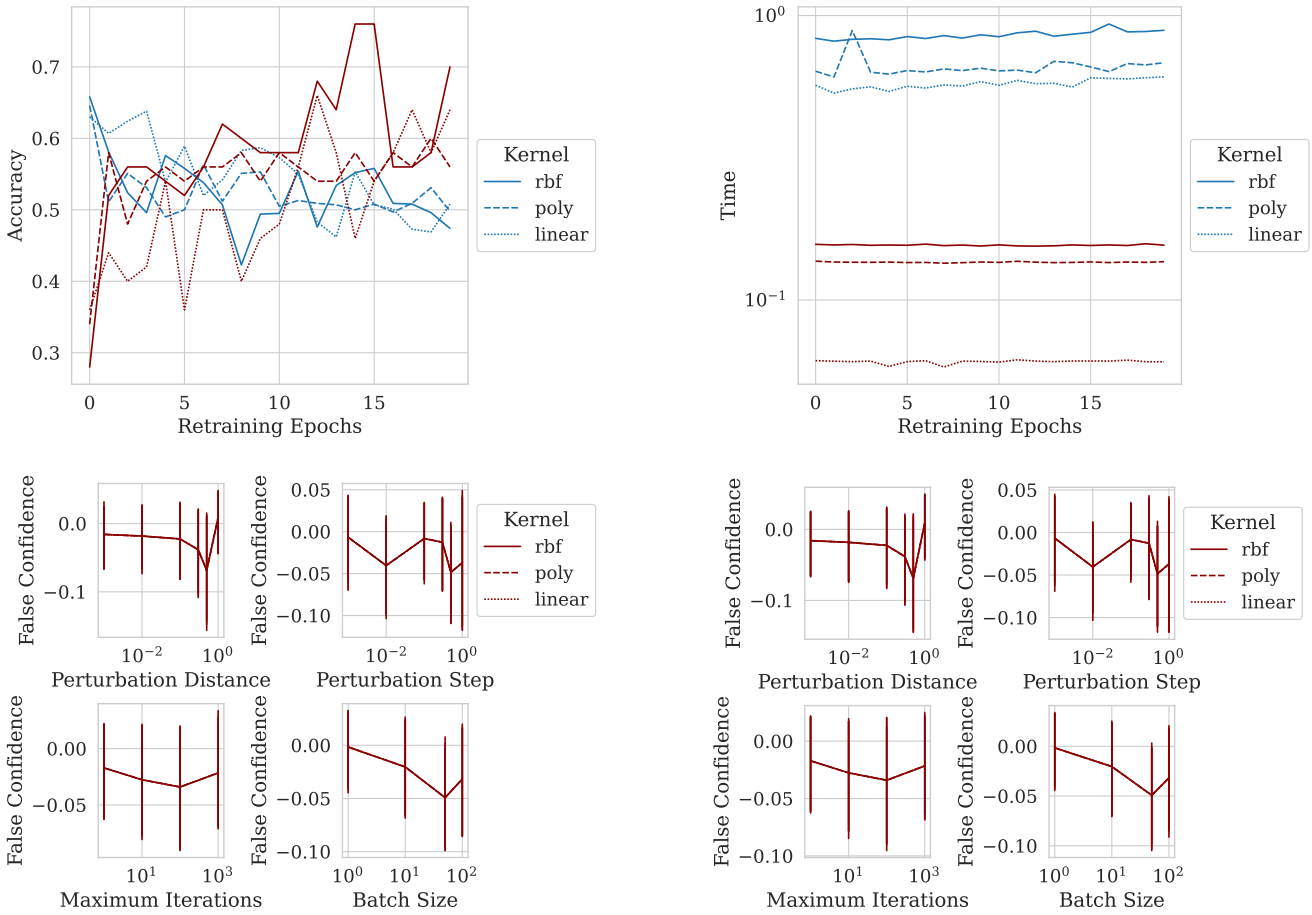


Figure B.2. Efficacy of Adversarial Retraining on Truthseeker Dataset. The top left depicts the adversarial and benign accuracy over a number of retraining epochs. The top right depicts the per epoch training time as the number of training epochs increases. The bottom row depicts the false confidence before retraining (left) on strong adversarial examples and after (right). The bars reflect the 95% confidence interval for all tested configurations.