The Space of Adversarial Strategies
Ryan Sheatsley, Blaine Hoak, Eric Pauley, and Patrick McDaniel,
University of Wisconsin-Madison
https://www.usenix.org/conference/usenixsecurity23/presentation/sheatsley
The Space of Adversarial Strategies

Ryan Sheatsley*
University of Wisconsin-Madison

Blaine Hoak*
University of Wisconsin-Madison

Eric Pauley
University of Wisconsin-Madison

Patrick McDaniel
University of Wisconsin-Madison

Abstract

Adversarial examples, inputs designed to induce worst-case behavior in machine learning models, have been extensively studied over the past decade. Yet, our understanding of this phenomenon stems from a rather fragmented pool of knowledge; at present, there are a handful of attacks, each with disparate assumptions in threat models and incomparable definitions of optimality. In this paper, we propose a systematic approach to characterize worst-case (i.e., optimal) adversaries. We first introduce an extensible decomposition of attacks in adversarial machine learning by atomizing attack components into surfaces and travelers. With our decomposition, we enumerate over components to create 576 attacks (568 of which were previously unexplored). Next, we propose the Pareto Ensemble Attack (PEA): a theoretical attack that upper-bounds attack performance. With our new attacks, we measure performance relative to the PEA on: both robust and non-robust models, seven datasets, and three extended $\ell_p$-based threat models incorporating compute costs, formalizing the Space of Adversarial Strategies. From our evaluation we find that attack performance to be highly contextual: the domain, model robustness, and threat model can have a profound influence on attack efficacy. Our investigation suggests that future studies measuring the security of machine learning should: (1) be contextualized to the domain & threat models, and (2) go beyond the handful of known attacks used today.

1 Introduction

It is well-known that machine learning models are vulnerable to adversarial examples—inputs designed to induce worst-case behavior. Seminal papers have introduced a suite of varying techniques for producing adversarial examples, each with their own unique threat models, strengths, and weaknesses [7,22,33,35,39]. Every generation of research yields the next evolution of attacks, designed to overcome prior defenses. It is unclear whether this evolution will ever converge, yet it is apparent that there are some attacks that have “survived” modern defenses. Specifically, the accepted baselines for evaluating defenses are converging to a small set of largely fixed attacks and threat models.

This observation on the fixed nature of commonly used attacks speaks to a broader and more fundamental problem in the way we evaluate the trustworthiness of machine learning systems: our understanding of adversaries has been derived from a union of works with disjoint assumptions and underlying threat models. As a consequence, it is challenging to draw any universal truths from a rather fragmented (and broadly incomparable) pool of knowledge. Subsequently, comparisons between attacks and attempts at characterizing the worst-case adversary have been through the lens of a specific threat model and defined with respect to a small handful of attacks, making it difficult to discern the true strength of claims on what is good (or even best) and when.

In this paper, we introduce a systematic approach to determine worst-case adversaries. We first introduce 568 new attacks by anatomizing seminal attacks into interchangeable components, therein enabling a meaningful evaluation of model robustness against an expansive attack space. With this decomposition, we formalize an extensible Space of Adversarial Strategies: the set of attacks considered by an adversary under a specific threat model and domain. We then empirically approximate the Pareto Ensemble Attack (PEA): a theoretical attack which upper-bounds attack performance by returning the optimal set of adversarial examples for a given threat model and dataset. We then use the PEA to explore a fundamental question: Does an optimal attack exist?

Our analysis begins by decomposing seminal attacks in adversarial machine learning. We observe that all known attacks are broadly built from two components: (1) a surface, and (2) a traveler. Surfaces encode the traversable attack space (often as the gradient of a cost function), while travelers are “vehicles” that navigate a surface to meet adversarial goals. Attack components live within surfaces and travelers, which characterize attack behavior, such as building crude surfaces that favor meeting adversarial goals without regard to budget,
or vice-versa. Our decomposition allows us to (a) generalize attacks in an extensible manner, and (b) naturally construct new (and known) attacks by permuting attack components.

From our decomposition, we permute attack components to build a previously unexplored attack space, yielding 568 new attacks. We then measure attack performance through the Pareto Ensemble, which is built by forming the lower envelope of measured model accuracy across attacks over the budget consumed. In other words, the Pareto bounds the performance an individual attack could achieve. We rank attacks with respect to the Pareto by measuring the difference in areas of their performance curves. Our approach not only gives us a comparable definition of optimality, but also a mechanism by which we can measure the merit of individual attacks.

Our evaluation across seven datasets, three threat models, and robust (through adversarial training) versus non-robust models found relative attack performance to be highly contextual. Specifically, (1) the domain and threat model can have a profound effect (especially if the trained model is robust), and (2) even the advantage of certain component choices is sensitive to these factors, as well as other paired components. We make the following contributions:

- We propose a decomposition of attacks in adversarial machine learning by atomizing attack components into two main layers, surfaces and travelers. Our decomposition readily enables extensions of new components.
- We characterize the attack space by permuting components of known attacks, yielding 568 new attacks.
- We introduce a systematic approach to compare the efficacy of attacks. We first build the Pareto Ensemble Attack from the performance curves of attacks and rank their relative performance.
- We instantiate and enumerate over a hypothesis space to identify which strategies perform better than others under a given threat model.

2 Background

2.1 Threat Models

Adversaries have historically had one of two goals: minimizing model accuracy [32, 35, 39, 54] or maximizing model loss [3, 22, 33]. The risks associated with minimizing model accuracy are often exemplified by vehicles misclassifying traffic signs [19], intrusion detection systems permitting malicious entities entry [60], medical misdiagnoses [20], among other failures. Maximizing model loss serves two purposes: (1) it is a surrogate for minimizing model accuracy (as, the inverse is performed to maximize model accuracy during model training), and (2) it aids in transferability attacks [18, 37, 38, 56]. In this work, we focus on minimizing model accuracy and defer the explorations of transferability to future work.

In the context of minimizing model accuracy, translating the risks above into an optimization objective to be solved by an adversary is commonly written as:

\[
\begin{align*}
\arg \min_{\varepsilon} & \quad \|\varepsilon\|_p \\
\text{subject to} & \quad f(x + \varepsilon) \neq \hat{y}, \\
& \quad x + \varepsilon \in B_{\phi}(x).
\end{align*}
\]

where we are given a victim model \(f\), a sample \(x\), label \(\hat{y}\), a self-imposed budget \(\phi\) measured under some \(\ell_p\)-norm. Conceptually, the adversary searches within some self-imposed norm-ball \(B\) of radius \(\phi\), centered at \(x\) for a “small” change \(\varepsilon\) that, when applied to \(x\), yields the desired goal.

With adversarial goals and capabilities defined, the final component of threat models pertains to access. Specifically, subsequent works have shown that adversaries need not have direct access to the victim model \(f\) to produce adversarial examples; models trained on similar data have similar decision manifolds, and thus, adversarial examples can “transfer” from one model to another [37, 38, 56]. When access is restricted (and thus, transferability is exploited), such threat models are called “grey-” or “black-box”, while full access to the victim model is called a “white-box” threat model. In this paper, we focus on white-box threat models as they represent the worst-case adversaries (in that they can produce adversarial examples with the tightest \(\ell_p\)-norm constraints). However, our decomposition and performance measurements can be directly applied to grey- and black-box threat models as well, which we further discuss in section 6.

On \(\ell_p\)-norms. As shown in Equation 1, the “cost” for crafting adversarial examples has been predominantly measured through \(\ell_p\)-norms. Informally, adversarial examples induce a misclassification between human and machine; \(\ell_p\)-bounded examples attempt to meet this definition. This concept arose from attacks on images, in that attacks would produce adversarial examples whose perturbations were invisible to humans, yet influential on models. \(\ell_p\)-norms are becoming an increasingly controversial topic, in that it has been debated if they have meaningful interpretations in non-visual domains [48], or even visual domains [10], or if they are useful at all [47]. Regardless, attacks have broadly converged on optimizing under \(\ell_0\), \(\ell_2\), or \(\ell_{\infty}\), and thus we focus our study on those.

2.2 Attack Algorithms

Here we briefly discuss the attack algorithms used in our decomposition (specifically, the unique components they introduce). We study these algorithms specifically due to their prevalence across works in adversarial machine learning [42].

Basic Iterative Method (BIM). BIM [29] is an iterative extension of Fast Gradient Sign Method (FGSM) [22]. BIM is an \(\ell_{\infty}\)-based attack that perturbs based on the gradient of a cost function, typically Cross-Entropy (CE). It often uses Stochastic Gradient Descent (SGD) as its optimizer for finding adversarial examples.
Projected Gradient Descent (PGD). PGD \cite{meng2020robust} is widely regarded as the state-of-the-art in crafting algorithms. PGD is identical to BIM, with the exception of a Random-Restart preprocessing step, wherein inputs are initially randomly perturbed within an $\ell_\infty$ ball.

Jacobian-based Saliency Map Approach (JSMA). The JSMA \cite{papernot2016limitations} is an $\ell_0$-based attack that is unique in its definition of a saliency map; a heuristic applied to the model Jacobian to determine the most salient feature to perturb in a given iteration. Unlike most other attacks, it does not rely on a cost function, but rather uses the model Jacobian directly. In our decomposition, we denote the JSMA saliency map as $SM_0$. The JSMA uses SGD as its optimizer.

DeepFool (DF). DF \cite{moosavi2016deepfool} is an $\ell_2$-based attack which models crafting adversarial examples as a projection onto the decision boundary. We find that we can model this projection as a saliency map, much like the JSMA, which we denote as $SM_\infty$. Similar to the JSMA, DF relies on the model Jacobian, does not have a cost function, and uses SGD.

Carlini-Wagner Attack (CW). CW \cite{carlini2017towards} is an $\ell_2$-based attack that is unique across several dimensions: (1) it uses a custom loss, which we label Carlini-Wagner Loss (CWL), (2) introduces the Change of Variables technique, which ensures that, during crafting, the intermediate adversarial examples always comply with a set of box constraints, and (3) uses Adam as its optimizer for finding adversarial examples.

AutoAttack (AA). AA \cite{liao2018defense} is an ensemble attack consisting of three different white-box attacks (as well as one black-box attack). This ensemble is unique in that all of its attacks are parameter free (except for the number of iterations to run attacks for). Its white-box attacks are: (1) Auto Projected Gradient Descent - Cross Entropy (APGD-CE), which is PGD with the Momentum Best Start optimizer, (2) Auto Projected Gradient Descent - Difference of Logits Ratio (APGD-DLR), which is APGD-CE but with Difference of Logits Ratio Loss, and (3) Fast Adaptive Boundary Attack \cite{katz2017reluplex} (FAB), which is similar to DeepFool, but its optimizer Backward Stochastic Gradient Descent applies a biased gradient step and a backward step to stay close to the original point.

### Table 1: Attack Component Decomposition.

<table>
<thead>
<tr>
<th>Surface Components</th>
<th>Traveler Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Losses:</td>
<td>Random-Restart:</td>
</tr>
<tr>
<td>Cross-Entropy</td>
<td>Enabled, Disabled</td>
</tr>
<tr>
<td>Carlini-Wagner Loss</td>
<td></td>
</tr>
<tr>
<td>Identity Loss</td>
<td></td>
</tr>
<tr>
<td>Difference of Logits Ratio Loss</td>
<td></td>
</tr>
<tr>
<td>Salience Maps:</td>
<td></td>
</tr>
<tr>
<td>$SM_0$, $SM_\infty$, $J$</td>
<td></td>
</tr>
<tr>
<td>Change of Variables</td>
<td></td>
</tr>
<tr>
<td>Optimizer:</td>
<td></td>
</tr>
<tr>
<td>$\ell_0$, $\ell_\infty$, $\ell_\infty$</td>
<td></td>
</tr>
<tr>
<td>$\ell_2$-norm</td>
<td></td>
</tr>
<tr>
<td>$\ell_0$-norm</td>
<td></td>
</tr>
<tr>
<td>$\ell_\infty$-norm</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Flow of composition between the surface and traveler to construct an attack. Required components have bold outlines while optional components have dotted outlines.

Importantly, these components are broadly mutually compatible with one another. In that one could omit, add, or swap them when building an attack. We exploit this property when permuting components, therein yielding a vast space of attacks, some of which are known, but most of which are not. This modular view of attacks not only allows us to build this vast space, but also makes the framework highly extensible by nature; new attacks can add on new choices for components or even new components entirely. A summary of the evaluated components in this paper and the compositions of well-known attacks are shown in Table 1.

For the remainder of this section, we describe: (1) the components that constitute a surface and their options, (2) the layers that define a traveler and associated configurations, and (3) a characterization of the attack space. An overview of the composition of the surface and traveler, and their interaction is shown in Figure 1. All symbols defined in this section (and in the remainder of the paper) can be found in appendix A.

### 3.1 Surfaces

Surfaces, which encode the traversable attack space, are built from: (1) the model Jacobian, (2) the gradient of a loss func-

---

**Project an image here**

---

**Figure 1: Flow of composition between the surface and traveler to construct an attack. Required components have bold outlines while optional components have dotted outlines.**
tion, (3) the application of a saliency map, and (4) an \ell_p\text{-norm context of crafting adversarial examples.}

Model Jacobian. At the heart of every surface (and thus, every attack) is the model Jacobian. The Jacobian \(J\) of a model with respect to a sample \(x\) encodes the influence each feature \(i\) in \(x\) has over each class. While most attack papers encode perturbations as a function of the gradient of a loss function, such computations necessarily involve computing a portion (at least) of the model Jacobian (whether attacks require the full model Jacobian is a matter of design choice). This is evident via application of the chain rule:

\[
\frac{\partial L(f(x), \hat{y})}{\partial x} = \frac{\partial L(f(x), \hat{y})}{\partial f(x)} \cdot \frac{\partial f(x)}{\partial x} = \frac{\partial L(f(x), \hat{y})}{\partial f(x)} \cdot J
\]

Importantly, computing a Jacobian is computationally expensive, on the order of \(O(d \cdot c)\), where \(d\) describes the dimensionality of \(x\) (i.e., the number of features) and \(c\) describes the number of classes. Thus, attacks that require the full model Jacobian (e.g., JSMA and DF) must pay a (sometimes substantial) cost in compute resources to produce adversarial examples—a fact largely overlooked. This component is perhaps the one with the greatest potential for extensibility. For instance, black-box attacks or those wanting to overcome obfuscated gradients [2] could opt to use Backwards Pass Differentiable Approximation (BPDA) [2] to obtain a jacobian rather than a traditional backwards pass.

Loss Functions. Perhaps the most popular design choice in attack algorithms is to perturb features based on the gradient of a loss function. The intuition is straightforward: we rely on surrogate measurements to learn parameters that have maximal accuracy during training, and thus, we can exploit these same measures to produce samples that induce minimal accuracy. This is commonly Cross-Entropy (CE) loss:

\[
\sum_{i} -\hat{y}_i \cdot \log(y_i)
\]

where \(c\) is the number of classes, \(\hat{y}_i\) is the label as a one-hot encoded vector, and \(y_i\) is the output of the softmax function.

Aside from CE loss, other attack philosophies instead opt for custom loss functions that explicitly encode adversarial objectives, such as Carlini-Wagner Loss (CWL):

\[
\|\delta\|_p^p - c \cdot \max(f_i(x) - \max\{f_i(x) : i \neq \hat{y}\})
\]

where \(p\) is the target \(\ell_p\text{-norm to optimize under and } c\) is a hyperparameter that controls the trade-off between the distortion introduced and misclassification.

Similar to the latter half of CWL, the Difference of Logits Ratio Loss (DLR) takes the difference between the true logit and the largest non-true-class logit. However, this loss function also divides by the difference between the largest logit (\(f_{\text{p}}(x)\)) and the third largest logit (\(f_{\text{p}}(x)\)), as follows:

\[
\frac{f_j(x) - \max\{f_i(x) : i \neq \hat{y}\}}{f_{\text{p}}(x) - f_{\text{p}}(x)}
\]

Finally, some attacks do not have an explicit loss function (such as JSMA or DF) and instead rely on information at other layers in the surface to produce adversarial examples (e.g., through saliency maps). To support this generalization, we implement a pseudo-identity loss function, Identity Loss (IL), which simply returns the \(\hat{y}\text{th model logit component.}

Saliency Maps. Saliency maps, in the context of adversarial machine learning, were first introduced by the JSMA [39]. These maps encode heuristics to best achieve adversarial goals by coalescing model Jacobian information into a gradient. We slightly tweak the original definition of the saliency map used in the JSMA to be: (1) independent of perturbation direction, and (2) agnostic of a target class. Though functionally different, we call this saliency map the Jacobian Saliency Map (SMJ), as the underlying heuristic is identical in spirit to the one introduced by the JSMA:

\[
SM_{J}(\hat{y}, J) = \begin{cases} 0 & \text{if } sgn(J_{\hat{y},i}) = sgn(\sum_{j \neq \hat{y}} J_{j,i}) \\ |J_{\hat{y},i}| \cdot \sum_{j \neq \hat{y}} |J_{j,i}| & \text{otherwise} \end{cases}
\]

where \(\hat{y}\) is the label for a sample \(x\), \(J\) is the Jacobian of a model with respect to \(x\), and \(i\) is the \(i\text{th feature of } x\). Moreover, we observe that attack formulations with complex heuristics, such as DeepFool, can be cast as-is into a saliency map as well. We define the DeepFool Saliency Map (SMF) as:

\[
SM_{F}(x, \hat{y}, q) = \frac{|f_j(x) - f_k(x)|}{\|J_y - J_k\|_q^q \cdot (J_y - J_k)^{q-1}} \cdot sgn(J_y - J_k)
\]

where \(x\) is a sample, \(\hat{y}\) is the label for \(x, q\) is calculated from the \(\ell_p\text{-norm where } q = \frac{p}{p-1}\), \(f\) is the model, and \(k\) is the “closest” class to the true label \(\hat{y}\) calculated by:

\[
k = \arg\min_{i \neq \hat{y}} \frac{|f_j(x) - f_i(x)|}{\|J_y - J_i\|_q^{q}}
\]

Notably, unlike the SMJ, this formulation is identical to that presented in the original DeepFool attack.

Finally, attacks can also opt not to use any form of saliency map, and thus, we define an identity saliency map, Identity Saliency Map (SMI), which simply returns the passed-in gradient-like information as-is.

\(\ell_p\text{-norms.} \) To meet threat model constraints, nearly all attacks manipulate gradient information via an \(\ell_p\text{-norm. We remark that this can be conceptualized as a layer in a surface. Thus, we provide abstractions for three popular }\ell_p\text{-based threat models:}

\[
\ell_p(\nabla) = \begin{cases} sgn(\nabla) & \text{if } i = \arg\max (|\nabla|) \\ 0 & \text{otherwise} \end{cases}
\]

where \(\nabla\) is some gradient-like information. While any \(\ell_p\text{-norm could be used in this layer, we also see natural extensions to other measurements of distance, such as LPIPS [62]...
that could also fit into this component. This layer could also extend to allow for adaptive threat models, such what is used in the DDN attack [44].

3.2 Travelers

Travelers serve as the “vehicles” that navigate over a surface to meet adversarial goals. Travelers are built from a series of subroutines that modify $x$: (1) Random-Restart, (2) Change of Variables, and (3) an optimization algorithm. Here, we detail these components and describe how they aid in finding effective adversarial examples.

Random-Restart. Many optimization problems, such as $k$-means [24] and hill-climbing [45], have been shown to benefit from the meta-heuristic, Random-Restart. Due to non-linear activation functions, deep neural networks are non-convex, and thus, subject optimization algorithms to non-ideal phenomena. Specifically, Random-Restart attempts to prevent optimization algorithms from becoming stranded in local minima by applying a random perturbation to an input. At this time, PGD is unique in its use of Random-Restart, defined as:

$$x = x + \mathcal{U}(-\varepsilon, \varepsilon)$$

where $\mathcal{U}$ is a uniform distribution, bounded by a hyperparameter $\varepsilon$ (which represents the total perturbation budget). Notably, while Random-Restart could be applied at each perturbation iteration, PGD uses it once on initialization.

Change of Variables. As a new way of enforcing box constraints, [7] introduced Change of Variables for the Carlini-Wagner Attack. As noted in [7], common practice for images is to first scale features to be within $[0, 1]$. When a perturbation is applied, these constraints must be enforced, as any feature beyond 1.0, for example, would map to a pixel value greater than 255, which exceeds the valid pixel range for 8 bit images. Most attacks enforce this constraint by simply clipping perturbations. However, this can negatively affect certain gradient descent approaches [7]. Thus, Change of Variables was proposed to alleviate deficient behaviors. In the context of CW, a variable $w$ is defined and solved for (instead of the perturbation directly). Its relationship to $x$ is:

$$x + \delta = \frac{1}{2}(\tanh(w) + 1)$$

where $\delta$ is the resultant perturbation applied to an input $x$. As [7] notes, this ensures that $0 \leq x + \delta \leq 1$, meaning that examples will automatically fall within the valid input range.

Optimizers. Nearly all attacks are described as “taking steps in the direction” (of a cost function). Practically speaking, these attacks refer to Stochastic Gradient Descent (SGD). As demonstrated by the BIM, as little as three iterations (with $\alpha = 0.01$) could be sufficient to drop state-of-the-art ImageNet models to $\sim 2\%$ accuracy [29]. However, Carlini-Wagner Attack was perhaps the first attack to explicitly use Adam to craft adversarial examples. Adam, unlike SGD, adapts learning rates for every parameter, and thus, often finds adversarial examples quicker than SGD [7].

In addition to SGD and Adam, we explore two additional optimizers, both of which come from AA. The first is Momentum Best Start (MBS), which accounts for momentum in its update step as follows:

$$x_{i+1} = x_i + \eta \cdot \alpha \cdot \delta_i + (1 - \eta) \cdot (x_i - x_{i-1})$$

where $\eta$ controls the strength of the momentum (set to 0.75 in [15]). In addition to this momentum step, it also features an adaptive learning rate that updates based on conditions that capture progression of inputs toward adversarial goals, described in [15].

Finally, our framework also supports Backward Stochastic Gradient Descent (BWSGD), which is the optimizer used for FAB in [13]. This optimizer operates similarly to SGD and MBS but aims to update with the distance to the original sample in mind by updating as follows:

$$x_{i+1} = x_i + (1 - \eta) \cdot \alpha \cdot \delta_i + \eta \cdot (x_{org} + \alpha \cdot \delta_{org})$$

where $\eta$ controls the influence of the original point on the update step. In addition, if $x_i$ is misclassified, this optimizer also performs a backward step by moving $x_i$ closer to $x_{org}$ via:

$$x_{i+1} = \beta \cdot x_{i+1} + (1 - \beta) \cdot x_{org}.$$  
In our experiments, we set $\eta$ to 0 since $\delta_{org}$ (a) does not translate to attacks that do not use a decision hyperplane projection and (b) as can be seen in [13], the use of backward step had a far greater influence on the attack performance than setting a non-zero value of $\eta$.

4 Extending Performance Measurements

With the attack space made enumerable by our decomposition, we now focus on necessary extensions of budget interpretations, the introduction of Pareto Ensemble Attack, and our approach for measuring optimality.

4.1 Beyond the $\ell_p$-norm

Since the inception of modern adversarial machine learning, the cost of producing an adversarial example has predominantly been measured through $\ell_p$-norms. Yet, it seems impractical to assume realistic adversaries will be unbounded by compute (as attacks that require days to produce adversarial examples offer little utility in any real-time environment). This observation is further exacerbated when attacks use expensive line search strategies [54], embed hyperparameter optimization as part of the crafting process [7], or rely on model Jacobian information [35, 39]. While such constructions can lead to incredibly effective attacks, adversaries who are limited by compute resources may find such attacks cost-prohibitive. To this end, we are motivated to extend standard definitions of budget beyond $\ell_p$-norms. Specifically, we incorporate and measure the time it takes to produce adversarial examples, therein extending our definition of budget as:

$$B(p, \theta, x) = \ell_p(x) + \theta \cdot T(x)$$ (2)
while adversaries with strong compute may not consider time (in other words, the Pareto frontier). Thus, we conclude that the optimal attacks best meet the adversarial goal, within the specified budget (i.e., the Pareto frontier). The

3750 32nd USENIX Security Symposium

5 Evaluation

Figure 2: The Optimal Attack — The PEA lower-bounds all attacks across the range of budgets. The area between the PEA and attack curves are shown with vertical bars.

where \( p \) is the desired norm, \( \theta \) parameterizes the importance of computational cost versus the introduced distortion, \( x \) is the adversarial example, and \( T \) returns the compute time necessary to produce \( x \). We note that the precise value of \( \theta \) depends on the threat model; adversaries who are compute-constrained may prioritize time twice as much as distortion (i.e., \( \theta = 2 \)), while adversaries with strong compute may not consider time at all (i.e., \( \theta = 0 \), as is done in standard evaluations). In section 5, we find that some attacks consume prohibitively large amounts of budget when compute is measured, and thus, current threat models (which only measure \( \ell_p \) distance) fail to generalize adversarial capabilities.

4.2 Pareto Ensemble Attack

With a realistic interpretation of budgets, we revisit a fundamental question: Does an optimal attack exist? Attacks measure distortion through different \( \ell_p \) norms, can require different amounts of compute, and have varying budgets (which is notably true for robustness evaluations). Thus, answering this question is non-trivial, especially in the absence of any meaningfully large attack space.

A single definition that accurately characterizes optimality across attacks, while incorporating these confounding factors, is challenging. Yet, we can say some attack \( A \) is optimal if, for a given threat model, \( A \) bounds all other attacks for an adversarial goal (i.e., \( A \) must lower-bound all attacks when minimizing model accuracy across budgets). Of the 576 attacks that we evaluated, no single attack met this definition. Thus, we conclude that the optimal attack are best characterized by an ensemble of attacks.

To this end, we introduce the Pareto Ensemble Attack (PEA), a theoretical attack which, for a given budget and adversarial goal, returns the set of adversarial examples that best meet the adversarial goal, within the specified budget (in other words, the Pareto frontier). The PEA is attractive for our analysis, in that it serves as a meaningful baseline from which we can compare attack performance to (discussed in the following section). We formally define the PEA as:

\[
\text{PEA} = \bigcup_{b \in \mathcal{B}} \left\{ \arg\min_{x' \in \mathcal{A}} \text{Acc}(f(x'), \hat{y}) \mid B(p, \theta, x') \leq b \right\}
\]

where \( b \) is a budget in a list of budgets \( \mathcal{B} \), \( x_A \) is the set of adversarial examples produced by attack \( A \) from a space of attacks \( \mathcal{A} \), \( f \) is a model, \( \hat{y} \) is the set of true labels for \( x_A \), \( B \) is a function used to measure budget (i.e., Equation 2), \( \text{Acc} \) returns model accuracy, \( p \) is an \( \ell_p \)-norm, and \( \theta \) controls the sensitivity to computational resources. Concisely, the PEA returns the set of adversarial examples whose model accuracy is minimal and within budget. Moreover, we provide a visualization of the PEA in Figure 2, where the PEA forms the lower envelope of model accuracy across budgets. We highlight that if there was some attack \( A' \) which achieved the lowest accuracy across all budgets (for some domain), then the PEA = \( A' \). It has been suggested by some in the community that algorithms such as PGD might be optimal for some application [5,33,63]. Our formulation of the PEA and measure of optimality allows us to test this hypothesis.

Measuring Optimality. The PEA yields a baseline from which we can fairly assess the performance of attacks. As the PEA meets the definition of optimal (that is, it bounds attack performance), we can evaluate attack performance relative to the PEA. Intuitively, attacks that closely track the PEA are performant, while those that do not are suboptimal. Mathematically, this can be measured as the area between the curves of the PEA and some attack \( A \). We note that our definition of optimality is: (1) relative to the attacks considered (and not measured against a set of provably worst-case adversarial examples or certified robustness [5,43,58]), and (2) as attacks are ranked by area, prefers attacks that are consistently performant (i.e., across the budget space). We acknowledge this measurement favors attacks whose behaviors are stable (which we argue most popular white-box attacks exhibit); other modalities may benefit from other cost measures.

For example, in Figure 2, the area between the PEA and attack \( A_2 \) is maximal, minimal for attack \( A_3 \), and somewhere in between for attack \( A_1 \). Thus, we conclude that the worst-case adversary would use \( A_3 \) if bound by small budgets, otherwise \( A_1 \) (and never \( A_2 \)). This approach to measuring attack performance is desirable in that, (1) attacks that track the PEA across budgets have minimal area (and thus, constitute a performant attack), and (2) attacks that are exclusively optimal for specific budgets incur large area.

5 Evaluation

With our attack decomposition and approach to measure optimality, we ask several questions: (1) Do known attacks per-
form best? (2) What attacks are optimal, if any? (3) Which components tend to yield performant attacks?

5.1 Setup

We perform our experiments on a Tensor EX-TS2 with two EPYC 7402 CPUs, 1 TiB of memory, and four Nvidia A100 GPUs. We use PyTorch [40] 1.9.1 for instantiating learning models and our attack decomposition. Here, we describe the attacks, threat models, robustness approach (i.e., adversarial training), and datasets used in our evaluation. We defer attack adaptations to appendix A.1, and hyperparameters & details on adversarial training to appendix A. Full versions of any shortened tables and figures in this section can be found in the arxiv version of this paper [49].

Attacks. In section 3, we introduce a decomposition of adversarial machine learning by atomizing attacks into modular components. Our evaluation spans the enumerated 576 attacks. Of these 576, JSMA, CW, DF, PGD, BIM, APGD-CE, APGD-DLR, and FAB are labeled explicitly, while other attacks are numbered from 0 to 755. The specific component choices of attacks mentioned by number can be found in appendix A. We note that some known attacks (such as DF and CW) have specialized variants for \( \ell_{p\neq 2} \)-norms, which we do not implement (as to maintain homogeneous behaviors across attacks of the same norm). Thus, we still reference these attacks numerically, since they are not the algorithmically identical.

In our experiments, we focused on untargeted attacks: that is, the adversarial goal is to minimize accuracy. While our decomposition is readily amenable to targeted variants, we defer analysis (and thus evaluation) of targeted attacks for two reasons: (1) choosing a target class requires domain-specific justification, and (2) certain classes are harder to attack than others [39]. These two factors would require a rather nuanced analysis, while our objectives aim to characterize broad attack behaviors. Thus, we anticipate that while a targeted analysis might affect attack performance in an absolute sense, relative performance to other attacks will likely be indifferent.

Threat Models. As motivated in section 4, we explore the interplay in attack performance when compute is measured, as defined by Equation 2. Specifically, we explore 3 \( \ell_p \)-based threat models (i.e., \( \ell_0, \ell_2, \) and \( \ell_{\infty} \)) with 20 different values of \( \Theta \) at 0.1 step sizes, from 0 to 2. These values can be interpreted as an adversary who, for example, values computational speed twice as much over minimizing distortion (i.e., \( \Theta = 2 \)). We note that all attacks are instantiated within our framework, and thus, any implementation-specific optimizations that accelerate compute speed are leveraged uniformly across attacks.

Robust Models. Adversarial training [22, 33] is one of the most effective defenses against adversarial examples to date [5, 14, 46]. Given its popularity and compelling results, we are motivated to investigate the impact of robust models on relative attack performance. We adversarially train our models with a PGD-based adversary. We follow the same approach as shown in [33]: input batches are replaced by adversarial examples (produced by PGD) during training. For MNIST and CIFAR-10, hyperparameters were used from [33]; other datasets were trained with parameters which maximized the accuracy over benign inputs and adversarial examples. Additional hyperparameters can be found in appendix A.

5.1.1 Datasets

We use seven different datasets in our experiments, chosen for their variation across dimensionality, sample size, and phenomena. We provide details and basic statistics below.

Phishing. The Phishing [12] dataset is designed for detecting phishing websites. Features were extracted from 5000 phishing websites and 5000 legitimate webpages. It contains 48 features and 10 000 samples. Beyond its phenomenon, we use the Phishing dataset to investigate the effects of small dimensionality and training size on attack performance.

NSL-KDD. The NSL-KDD [55] is based on the seminal KDD Cup ’99 network intrusion detection dataset. Features are defined from varying network features from traffic flows emulated in a realistic military network. At 41 features, it contains 125 973 samples for training and 22 544 for testing. We use the NSL-KDD for its small dimensionality, large training size, and concept drift [21].

UNSW-NB15. The UNSW-NB15 [36] is a network intrusion detection dataset designed to replace the NSL-KDD. Features are derived from statistical and packet analysis of real innocuous flows and synthetic attacks. It has 48 features, with 175 341 samples for training and 83 332 samples for testing. The UNSW-NB15 enables us to compare if attacks generalize to similar phenomenon (such as the NSL-KDD).

MalMem. CIC-MalMem-2022 (MalMem) [8] is a modern malware detection dataset. 58 features are extracted from memory dumps of benign applications and three different malware families (i.e., trojans, spyware, and ransomware). In total, it contains 58,596 samples, with half belonging to benign applications and half to malware. MalMem gives us the opportunity to understand the effects of small dimensionality in an entirely different phenomenon from the network datasets.

MNIST. MNIST [30] is a dataset for handwritten digit recognition. It is a well-established benchmark in adversarial machine learning applications. With 784 features, 60 000 samples for training and 10 000 for testing, MNIST has substantially larger dimensionality than even the largest network datasets. We use MNIST to corroborate prior results, explore a vastly different phenomenon, and investigate how (relatively) large dimensionality influences attack performance.

FMNIST. Fashion-MNIST (FMNIST) [59] is a dataset for recognizing articles of clothing from Zalando articles. Advertised as a drop-in replacement for MNIST, FMNIST was designed to be a harder task and closer representative of modern
computer vision challenges. F MNIST has identical dimensionality, training samples, and test samples to MNIST. Thus, we use FMNIST to understand if changes in phenomena alone are sufficient to influence attack performance.

CIFAR-10. CIFAR-10 [28] is a dataset for object recognition. Like MNIST, CIFAR-10 is extensively used in adversarial machine learning literature. At 3072 features, CIFAR-10 represents a substantial increase in dimensionality from MNIST. It has 60,000 samples for training and 10,000 for testing. CIFAR-10 allows us to compare against extant works and explore how domains with extremely large dimensionality affect attack efficacy.

5.2 Comparison to Known Attacks

As discussed in section 3, we contribute 568 new attacks. Naturally following, we ask: are any of these attacks useful? Asked alternatively, do known attacks perform best? We investigate this question through commonly accepted performance measurements [6, 29, 35, 39]: the amount of $\ell_p$ budget consumed by attacks whose resultant adversarial examples cause model accuracy to be $< 1\%$. In this traditional performance setting, we aim to understand if known attacks serve as the Pareto frontier (which would indicate that our contributed attacks yield little in terms of adversarial capabilities).

We organize our analysis as follows: (1) we first segment attacks based on $\ell_p$-norm and compare them to known attacks of the same norm (that is, we compare JSMA to $\ell_0$ attacks, CW, DF, & FAB to $\ell_2$, and PGD, BIM, APGD-CE, & APGD-DLR to $\ell_\infty$), and (2) report relative budget consumed (with respect to known attacks) for attacks whose adversarial examples caused model accuracy to be $< 1\%$.

5.2.1 Performance on MNIST

For our analysis of attack performance, we craft adversarial examples for 1000 iterations over ten trials (we note that 1000 iterations was selected for completeness; the vast majority of attacks converged in less than 100 iterations). Figure 3 shows the median results for two threat models, segmented by $\ell_p$-norm. Known attacks (i.e., JSMA, CW, DF, FAB, PGD, BIM, APGD-CE, and APGD-DLR) are highlighted in red, while other attack curves are dotted blue and slightly opaque to capture density. We now discuss our results on a per-norm basis.

$\ell_0$ Attacks. Figure 3a shows $\ell_0$-targeted attack performance with the JSMA in red. We observe that the JSMA is worse than most attacks. Attack performance is largely well-clustered with a few poor performing attacks near the top right portions of the graph. These attacks used Random-Restart, and thus, immediately consume most of the available $\ell_0$ budget.

$\ell_2$ Attacks. Figure 3b shows $\ell_2$-targeted attacks, with CW as solid red, DF as dash-dotted red, and FAB as dashed red. Like $\ell_0$, attacks are well-concentrated (albeit with slightly more spread). Notably, DF and FAB (which are ostensibly superimposed on one another), demonstrate impressive performance (the red lines that are nearly vertical)—both drop model accuracy with a near-zero increase in budget. CW exhibits moderate performance over the budget space.

$\ell_\infty$ Attacks. Figure 3c shows $\ell_\infty$-targeted attacks, with PGD as solid red, BIM as dash-dotted red, APGD-CE as dashed red, and APGD-DLR as dotted red. Unlike other norms, $\ell_\infty$ has clear separation, broadly attributable to using Change of Variables (specifically, attacks that used Change of Variables performed worse than those that did not). Finally, all of the known attacks exhibit near-identical performance, with APGD-DLR slightly pulling ahead at budgets $> 0.2$.

From our norm-based analysis, we highlight that: (1) Random-Restart is largely inappropriate for $\ell_0$-targeted attacks (in that benefits do not outweigh the cost), (2) $\ell_2$-targeted attacks cluster fairly well; no individual attack substantially outperformed any other, and (3) $\ell_\infty$-targeted attacks were broadly unable to exploit Change of Variables.

5.2.2 Relative Performance to Known Attacks

Recall our central question for this experiment: do known attacks perform best? To answer this question, we analyze the minimum budget necessary for attacks to cause model accuracy to be $< 1\%$ (attacks that fail to do so are encoded as consuming infinite budget). We run attacks for 1000 iterations over ten trials and report the median results in Table 2. Here, attacks are ranked by budget and segmented by norm (i.e., 192 attacks per norm). We report the percentage change of each attack with respect to the known attack that performed best in that norm (that is, for $\ell_0$, results are relative to the JSMA, while for $\ell_2$, results are relative to CW, which outperformed DF, etc.). In the table, we show: (1) the attack that ranked first, (2) ranks of known attacks, and (3) the lowest ranked attack that still reduced model accuracy to $< 1\%$. Next, we highlight some strong trends for each $\ell_p$-norm.

Of the 34% of attacks that succeed in the $\ell_0$ space, the JSMA (ranked 32$^{nd}$) was at the bottom of the highly performant bin (in that its $\ell_0$ budget was 0.17)—the JSMA was held back by its saliency map, SM$_J$; using either SM$_D$ (or no saliency map at all, i.e., SM$_J$) was almost always better. While CW seemingly rank low (i.e., 69$^{th}$), we note that $\ell_2$ budgets were broadly similar, as the worst and best performing attacks were within $\pm 50\%$ of the budget consumed by CW, APGD-DLR, BIM, APGD-CE, and PGD, ranked 17$^{th}$, 40$^{th}$, 48$^{th}$, and 49$^{th}$ respectively, were marginally outperformed by attacks using either the SM$_D$ saliency map or BWSGD optimizer. As a final note, we were confounded by the performance of DF and FAB—visually inspecting Figure 3b, both are clearly superior attacks (the performance curves ostensibly resemble square waves) and yet, they failed to reduce model accuracy to $< 1\%$. While these analyses of attack performance has been useful historically for understanding adversarial examples, we argue
that this “race to 0 % accuracy” fails to capture meaningful definitions of attack performance (as made evident by the apparent “failure” of DF and FAB).

From our comparison with known attacks, we highlight two key takeaways: (1) Measuring the required distortion to reach some amount of model accuracy is a rather crude approach to estimating attack performance. We argue using measurements that factor the entire budget space (such as the PEA, which we use subsequently) will yield more meaningful interpretations of attack performance. (2) Even when we define success as <1 % model accuracy, known attacks do not perform best. In fact, many attacks produced by our decomposition consistently outperformed known attacks (e.g., 68 out of the 189 introduced by our approach outperformed both CW and DF), which demonstrates the novel adversarial capabilities introduced by our decomposition.

### 5.3 Optimal Attacks

In Figure 4.2, we introduced an approach for measuring optimality: the area between the performance curves of the PEA and an attack. Attacks that have a small area closely track the PEA and thus, are performant attacks, while those that have a large area perform poorly. In this experiment, we ask: does attack performance generalize? In other words, is relative attack performance invariant to dataset or threat model?

We investigate this hypothesis of attack optimality by ranking attacks by area across varying threat models, datasets, and robust models. Then, we measure the generalization of these rankings via the Spearman rank correlation coefficient [52], which informs us how similar the rankings are between two datasets, threat models, or a robust and non-robust model.

For example, a highly positive correlation across two datasets would imply that relative attack performance was unchanged (in other words, changing the dataset had little to no effect on attack performance), a near-zero correlation would suggest that relative attack performance changed substantially (which would suggest attack performance is sensitive to the dataset), and a negative correlation would indicate attack performance was reversed (i.e., the worst attacks on one dataset...
became the best on another). In our experiments, we craft adversarial examples for 1000 iterations over ten trials and report the median Spearman rank correlation coefficients. We note that 1000 iterations was selected for completeness; the vast majority of attacks converged in less than 100 iterations.

**Optimal Attacks by Threat Model.** Here, we analyze the generalization of attack performance across threat models. Specifically, we consider $\ell_0$, $\ell_2$, and $\ell_\infty$-based threat models with varying values of $\theta$ (from 0 to 2). Note that $\theta = 0$ (i.e., where compute time is ignored) is the commonly used threat model. Figure 4 shows the median Spearman rank correlation coefficients for MNIST with results segmented by $\ell_p$-norm. Each entry corresponds to a unique threat model (i.e., a value for $\theta$). High attack performance generalization is encoded as lighter shades, while low generalization is encoded with darker shades.

From the results, we can readily observe: (1) rankings do not generalize across $\ell_p$-norms, especially between $\ell_0$- and $\ell_\infty$-targeted attacks (but do generalize relatively well within an $\ell_p$-norm), and (2) the influence of compute on rankings appears to be $\ell_p$-norm dependent: $\ell_0$-based threat models that weight compute (i.e., $\theta \neq 0$) do not generalize well to those that do not, $\ell_2$-based threat models exhibit a smoother degradation of generalization, while, surprisingly, $\ell_\infty$-based threat models generalize everywhere (that is, the same attacks that were found to be performant with $\theta = 2$ were as performant when $\theta = 0$). Within a dataset, we observe that the threat model significantly affects attack performance across $\ell_p$-norms, and to some extent, within an $\ell_p$-norm, with $\ell_\infty$ as the exception.

**Optimal Attacks by Dataset.** In this experiment, we instead now measure the generalization of attack performance across datasets. Specifically, for a given threat model, we measure the generalization of attack performance rankings across seven datasets. The evaluated datasets span varying forms of phenomena, from classifying network traffic to categorizing clothing items, and thus, we investigate if performant attacks are task-agnostic.

The results in Figure 5 disclose that: (1) CIFAR-10 does not generalize at all, regardless of $\ell_p$-norm, (2) skewing budgets towards favoring compute gradually degrades generalization—attack rankings become increasingly dissimilar as we move from ignoring compute time ($\theta = 0$) to heavily favoring it ($\theta = 2$), and (3) $\ell_0$-based threat models generally generalize across datasets and is largely invariant to considering compute, $\ell_2$ attacks, regardless of $\theta$, moderately generalize, and $\ell_\infty$ attacks closely track $\ell_2$ attacks with a particular subtly: attacks performant on MNIST generalized almost perfectly to FMNIST (i.e., image-based generalization), while attacks performant on NSL-KDD almost perfectly generalized to UNSW-NB15 (i.e., network-intrusion-detection-based generalization). Lastly, we observe that attacks performant on Phishing and MalMem moderately generalized better to non-image data (particularly to the UNSW-NB15). Considering compute degrades these observations slightly. Within a threat model, we observe that, based on $\ell_p$-norm, the dataset can have drastic degrees of influence on attack performance, in that it can have little effect at all (e.g., $\ell_0$), have an effect everywhere (i.e., $\ell_2$), or have an effect specific to the phenomena (i.e., $\ell_\infty$). We attribute the unique behavior of CIFAR-10 to its dimensionality; the next largest dataset, FMNIST and MNIST, are $\approx 74\%$ smaller.

**Optimal Attacks Against Robust Models.** In this final experiment, we now measure the generalization of attack performance between robust and non-robust models. Specifically, for a given threat model and dataset, we compute pairwise correlations between attack performance rankings on robust and non-robust models. Adversarially trained models have been shown to be an effective defense against adversarial examples [33], and thus, we investigate if such procedures have a visible effect on attack performance.

Median Spearman rank correlation coefficients for all datasets and threat models are shown in Figure 6. We note several trends across norm, threat models, and datasets: (1) generally speaking, attack rankings in $\ell_2$-based threat models were substantially affected by robust models, especially for MalMem, MNIST, and FMNIST, (2) considering compute can have a significant impact on generalization, mainly dependant on the norm; increasing the importance of compute almost universally aided generalization in $\ell_2$, but hurt generalization in $\ell_\infty$ (especially for image data, albeit CIFAR-10 is less sensitive to varying $\theta$ at the scales we investigated), and (3) we observed that top-performing attacks can be especially affected: on MNIST for an $\ell_2 + 0$-time threat model, for instance, the top 10 attacks on the non-robust model had a median rank of 445th (out of 576) on the robust model. These profound differences in relative attack effectiveness demonstrate that the unique properties of robust models necessitate changes to attack components (discussed further in subsection 5.4.3).

**Takeaways on Attack Optimality.** In this set of experiments, we analyzed attack optimality through the lens of varying threat models, unique data phenomena, and robust models. From our analyses, we find that the optimality of any given attack is highly dependant on the given context. We support this conclusion through the following remarks on the generalization of relative attack performance: (1) across threat models, performance generalizes well within an $\ell_p$-norm, but not across—considering compute exacerbates this observation, (2) across datasets, performance generalization is broadly sensitive to $\ell_p$-norm (with CIFAR-10 generalizing poorly everywhere), and (3) between robust and non-robust models, attack rankings are largely a function of data phenomena (e.g., image-based phenomena exhibit poor generalization, regardless of the threat model).
5.4 When and Why Attacks Perform Well

With our metric for attack performance established and evaluated, we proceed by asking, why do certain attacks perform well? Here, we explore the general trends of attack components and their influence on performance through a series of hypothesis tests. We build a space of possible hypotheses of relative attack performance (over all attack components), perform hypothesis testing against this space, and identify those with the highest significance and effect size. We begin with significant hypotheses of non-robust models and conclude with hypotheses most affected by model robustness.

5.4.1 The Space of Hypotheses

We define a hypothesis as a comparison between two component values (which we label as $H_1$ and $H_2$), such as “using Cross-Entropy is better than Carlini-Wagner Loss.” Now, we want to understand the conditions that make a hypothesis true. These conditions can be using a specific dataset, under a certain threat model, or based on other component values. Building off our previous example, this hypothesis paired with a condition could be “using Cross-Entropy is better than Carlini-Wagner Loss, when the dataset is Phishing.” When we test a hypothesis, we look at the statistical significance of the hypothesis under all conditions to determine when a hypothesis is true. Enumerating across all possible hypothesis and condition pairs yielded 1690 candidate hypotheses. It should be noted that the component values in hypotheses are always in the same component, as comparing usefulness across components would be nonsensical (e.g., “Using Cross-Entropy is better than using Random-Restart” is not meaningful).

5.4.2 Testing

We test the 1690 hypotheses with the Wilcoxon Signed-Rank Test, a non-parametric pairwise test, equivalent to a pairwise Mann-Whitney $U$ Test, to determine its significance. We also report the effect size of the test, defined as the percentage of pairwise median areas (over ten trials, with trial counts factored into computed $p$-values) from component $H_1$ that were smaller than component $H_2$ (recall, a smaller area corresponds to a better attack, as it more closely tracks the PEA). Note that the $p$-values for many hypotheses underflowed 64 bit floating point precision, implying that the results of the test are highly significant across all datasets and threat models. A subset of hypotheses are represented in Table 3.

We find many highly-significant correlations in the results across the space of hypotheses. Specifically, we set a significance threshold proportional to the number of hypothesis tests we evaluated to minimize false positives:

\[ p < \frac{1}{1690} = 5 \times 10^{-6}. \]

We found that 1536 (90\%) of hypotheses were below this threshold. We highlight the most prominent conclusions among these 1536 hypothesis: (1) Change of Variables was found to be disadvantageous—86 hypotheses involving Change of Variables met our threshold; all 86 were against its use, (2) Adam was superior to all other optimizers—503 hypotheses comparing Adam to other optimizers met our threshold, of which 50\% of them ruled in favor of Adam (with SGD at 33\%, and MBS at 16\%), (3) Random-Restart was found to be preferable across 61\% of hypotheses (51 of 83), (4) $\ell_\infty$-targeted attacks, at 79\% (163 of 205) were superior to both $\ell_0$- and $\ell_2$-targeted (which were only favorable 16\% (34 of 205) and 4\% (8 of 205) of the time, respectively), (5) using no saliency map (i.e., $SM_1$) was better 70\% (131 of 187) of
When sorted by effect size, the top 50% of hypotheses have an effect size greater than 80%.

These hypothesis tests provide statistical evidence of some phenomena across our experiments. Specifically, we see large migrating from non-robust to robust models largely concern model (labeled as delta). Many of the top hypotheses when sorted by the change in effect size from a non-robust to robust models is, the largest changes in effect size) by robust models.

We highlight some key takeaways from this experiment: (1) These hypothesis tests provide statistical evidence of some common practices within the community (using Random-Restart and the superiority of Adam), while also demonstrating some perhaps surprising conclusions, such as the detriment of using Cross-Entropy over no loss function at all. (2) We emphasize the utility of hypothesis testing for threat modeling as well: the tests provide a schema for performing worst-case benchmarks in their respective domain. For example, when benchmarking MNIST against $\ell_0$-based adversaries, attacks that use the Jacobian Saliency Map are likely to outperform attacks that use DeepFool Saliency Map.

### 5.4.3 The Effect of Model Robustness

As shown in Figure 6, robust models can have a significant impact on attack rankings. Here, we investigate why such broad phenomena occur. Specifically, we investigate how attack parameter choices change performance on a robust versus a non-robust model. We repeat our hypothesis testing on robust models only and compare the hypotheses most affected (that is, the largest changes in effect size) by robust models.

Table 4 provides a listing of the top pairs of hypotheses, sorted by the change in effect size from a non-robust to robust model (labeled as delta). Many of the top hypotheses when migrating from non-robust to robust models largely concern CIFAR-10 and MalMem, which were broadly the most unique phenomena across our experiments. Specifically, we see large changes in losses and saliency maps for the attacks that were effective at attacking robust models. The emphasis on CE could be in part attributed to the fact that both the model is trained on this loss as well as used by PGD, the attack used to generate adversarial examples within minibatches. This observation suggests that attacks using losses also used in adversarial training are highly effective.

Beyond the influence of loss on CIFAR-10 and MalMem, most of our tested hypotheses remained relatively unaffected by model robustness: of the 1690 hypotheses tested, only 334 had an effect size change of 10% or greater between robust and non-robust models. This implies that, while many of the factors that make attacks effective do not vary between normally- and adversarially-trained models, the subset that does vary accounts for a vast difference in attack effectiveness.

<table>
<thead>
<tr>
<th>Component $H_1$</th>
<th>Component $H_2$</th>
<th>Condition</th>
<th>p-value</th>
<th>Effect Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. SGD</td>
<td>is better than</td>
<td>$\ell_2$-based Difference of Logits Ratio Loss</td>
<td>when Dataset = MNIST</td>
<td>$&lt;2.2 \times 10^{-6}$</td>
</tr>
<tr>
<td>2. Adam</td>
<td>is better than</td>
<td>$\ell_2$-based Difference of Logits Ratio Loss</td>
<td>when Dataset = MNIST</td>
<td>$&lt;2.2 \times 10^{-6}$</td>
</tr>
<tr>
<td>84. Identity Loss</td>
<td>is better than</td>
<td>$\ell_2$-based Difference of Logits Ratio Loss</td>
<td>when Dataset = NSL-KDD</td>
<td>$&lt;2.2 \times 10^{-6}$</td>
</tr>
<tr>
<td>85. SGD</td>
<td>is better than</td>
<td>$\ell_2$-based Difference of Logits Ratio Loss</td>
<td>when SaliencyMap = Jacobian Saliency Map</td>
<td>$&lt;2.2 \times 10^{-6}$</td>
</tr>
<tr>
<td>393. DeepFool Saliency Map</td>
<td>is better than</td>
<td>Jacobian Saliency Map</td>
<td>when Dataset = PHRIST</td>
<td>$&lt;5 \times 10^{-6}$</td>
</tr>
<tr>
<td>394. Cross-Entropy</td>
<td>is better than</td>
<td>Carlini-Wagner Loss</td>
<td>when Change of Variables = Disabled</td>
<td>$&lt;5 \times 10^{-6}$</td>
</tr>
<tr>
<td>1689. $\ell_0$</td>
<td>is better than</td>
<td>$\ell_2$-based Difference of Logits Ratio Loss</td>
<td>when Threat Model = $\ell_1 + 1.0$</td>
<td>$9.8 \times 10^{-1}$</td>
</tr>
<tr>
<td>1690. Identity Saliency Map</td>
<td>is better than</td>
<td>DeepFool Saliency Map</td>
<td>when Threat Model = $\ell_0 + 0.4$</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 3: The evaluated hypotheses for non-robust models. The top 344 hypotheses have a p-value that exhibits 64 bit underflow. When sorted by effect size, the top 50% of hypotheses have an effect size greater than 80%.

---

3One would expect evaluating 1000 hypotheses at $p < 0.01$ significance would result in 10 false positives, for example.
While this initial application of our framework focused on work is complementary in that we provide a broad, modular approach with black-box components to understand the trade-offs as there are a variety of techniques for efficiently mounting black-box adversaries, such as using Backward Pass Differentiable Approximation [2] in place of the model Jacobian, or introduced to combine the efficacy of AutoAttack with respect to common failures of defenses, it may fail on defenses where an expert-designed adaptive attack would succeed. Thus, the Adaptive AutoAttack (A^3) extension was introduced to combine the efficacy of AutoAttack, while dynamically adapting to new defenses [61]. A^3 frames building adaptive attacks as a search problem, wherein a surrogate model is built and a “backbone” attack (e.g., FGSM, PGD, CW, among others) is greedily selected, paired with a loss function and subroutines (such as Random-Restart). A^3 builds upon AutoAttack in that it enables searching through the attack design space to find the most effective adaptive attack. Our work is complementary in that we provide a broad, modular attack space, while A^3 provides an approach for building adaptive attacks dynamically.

### 7 Conclusion

In this paper, we introduced the space of adversarial strategies. We first presented an extensible decomposition of current attacks into their core components. We subsequently constructed 568 previously unexplored attacks by permuting these components. Through this vast attack space, we measured attack optimality via the PEA: a theoretical attack that upper-bounds attack performance. With the PEA, we studied how attack rankings change across datasets, threat models, and robust vs non-robust models. From these rankings, we described the space of hypotheses, wherein we evaluated how component choices conditionally impact attack efficacy. Our investigation revealed that attack performance is highly contextual—certain components can help (or hurt) attack performance when a specific \( \ell_p \) norm, compute budget, domain, and even phenomena is considered. The space of adversarial strategies is rich with highly competitive attacks; meaningful evaluations need to consider the myriad of contextual factors that yield performant adversaries.

### 8 Acknowledgements

We would like to thank the anonymous reviewers for their insightful feedback throughout the review cycle. Additionally, we would like to thank Quinn Burke, Yohan Beugin, Rachel King, and Dave Evans for their helpful comments on earlier versions of this paper.

This research was sponsored by the Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Combat Capabilities Development Command Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes not withstanding any copyright notation here on. This material is based upon work supported by the National Science Foundation under Grant No. CNS-1805310.
References


[48] Ryan Sheatsley, Blaine Hoak, Eric Pauley, Yohan Beugin, Michael J. Weisman, and Patrick McDaniel. On the


A Miscellany

Table 5 provides a listing of model hyperparameters for each of our datasets. Our selection was inspired by publications that report state-of-the-art accuracy for the models we used. Table 7 provides a listing of all symbols used in this paper and their associated meanings. Table 6 provides the parameters used for adversarial training. Finally, we provide Table 8 for translating attack numbers to component values.

A.1 Attack Modifications

Carlini-Wagner Attack. As described in section 3, the $\text{CW}$ attack loss function includes a hyperparameter $c$ which controls the trade-off between the distortion introduced and misclassification. In the original attack definition, $c$ is optimized dynamically through binary-search [7]. This is cost-prohibitive and prevents us from performing any meaningful evaluation when computational cost is considered (as this attack would exist on a separate scale, when compared to $\text{PGD}$ or even the JSMA, which requires the model Jacobian). To remedy this, we select a constant value of $c$ in our experiments. From the investigation on values of $c$ in [7] with respect to attack success probability versus mean $f_2$ distance, we choose a value of 1.0 for $c$ in all experiments.

Jacobian-based Saliency Map Approach. The original definition of the JSMA included a search space, which defined the set of candidate features to be selected for perturbation. In the original publication, the JSMA initially set $\alpha$ to either 1 or 0 (that is, pixels were fully turned “off” or “on”). We find that this underestimates the performance of the JSMA on many datasets. Instead, we derive a more effective strategy of instead setting the saliency map score for some feature $i$ to $0$ if: (1) the saliency score for $i$ is positive and $x_i = 1$, or (2) the saliency score for $i$ is negative and $x_i = 0$. This prevents our version of the JSMA from selecting features that are already at limits of valid feature values (i.e., 1 and 0). Moreover, we do not select pixel pairs, as described in [39], as we found our implementation to be at least as effective (often more) as the original JSMA.

Difference of Logits Ratio Loss. The original formulation of DLR requires takes the ratio of the differences between: (1) the true logit and largest non-true-class logit, and (2) the largest logit and the third largest logit. In our evaluation, we used datasets that had less than three classes. For those scenarios, we take the second largest logit.