SANDIA REPORT
SAND2020-1398
Printed September 2020

Sandia National Laboratories

Effects of Jacobian Matrix Regularization on the Detectability of Adversarial Samples

Michael Eydenberg, Kanad Khanna, Ryan Custer

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

Effects of Jacobian Matrix Regularization on the Detectability of Adversarial Samples

Michael Eydenberg
Computational Decision Science
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1027
mseyden@sandia.gov

Kanad Khanna
Special Analytic Initiatives
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1027
kkhanna@sandia.gov

Ryan Custer
Computational Decision Science
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1027
rpcuste@sandia.gov

SAND2020-13986

ABSTRACT

The well-known vulnerability of Deep Neural Networks to adversarial samples has led to a rapid cycle of increasingly sophisticated attack algorithms and proposed defenses. While most contemporary defenses have been shown to be vulnerable to carefully configured attacks, methods based on gradient regularization and out-of-distribution detection have attracted much interest recently by demonstrating higher resilience to a broad range of attack algorithms. However, no study has yet investigated the effect of combining these techniques. In this paper, we consider the effect of Jacobian matrix regularization on the detectability of adversarial samples on the CIFAR-10 image benchmark dataset. We find that regularization has a significant effect on detectability, and in some cases can make an undetectable attack on a baseline model detectable. In addition, we give evidence that regularization may mitigate the known weaknesses of detectors to high-confidence adversarial samples. The defenses we consider here are highly generalizable, and we believe they will be useful for further investigations to transfer machine learning robustness to other data domains.

ACKNOWLEDGMENT

The authors would like to thank Jeremy Wendt for his insights into machine learning security and methods for evaluating defenses. We would also like to thank the sponsor technical leadership team for their input and general support of this activity.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

AUROC	Area Under the Receiver Operating Characteristic
BA	Boundary Attack
BIM	Basic Iterative Method
CIFAR	Canadian Institute For Advanced Research image collection
CWL2	Carlini and Wagner L^2 optimization method
DDNET	Defensive-Distillation Network
DM	Deep Mahalanobis
FSGM	Fast Gradient Sign Method
JR	Jacobian regularization
LDA	Linear Discriminant Analysis
LID	Local Intrinsic Dimensionality
MNIST	Modified National Institute of Standards and Technology database of handwritten digits
OOD	Out-of-Distribution Detection
RESNET	Residual Network

1. INTRODUCTION

As machine learning systems become more widely used adversaries will attack them in order to achieve some goal. This is part of the natural progression of the “arms race” between attackers and defenders when a new technology is introduced. Machine learning systems are vulnerable to *adversarial examples*, which are deliberately altered inputs that force a machine learning system to misclassify the input. By forcing this misclassification, an attacker achieves some result such as avoiding detection, subverting model confidence, or recovering sensitive data. In general, adversarial examples are easy to generate and usually involve adding noise to a known good example, but they can also be constructed and do not have to be based on known input [31].

To keep up in the arms race, security professionals must develop new methods to defend machine learning systems against adversarial examples. This is a difficult task because where an attacker need only find a single adversarial example that achieves their goal, a security professional must be able to mitigate the effectiveness of, and ideally detect, every possible adversarial input.

In machine learning systems, this arms race began with adversaries making a single perturbation to an input that resulted in miscategorization. This quickly evolved into iterative gradient-based attacks that make repeated perturbations based on their effects on the target model. Gradient-based attacks work by using the output of a neural network’s classification layer to guide the creation of adversarial examples and allow adversaries to efficiently target their attacks [3]. In turn, security professionals began using defensive distillation and gradient obfuscation techniques to mask the gradient output by the neural network layers so adversaries had much less information to target their attacks. Although these results looked promising at first, they were quickly bypassed by adversaries and are now considered thoroughly defeated [3, 7, 8]. For a more formal definition of gradient based attacks, adversarial examples, and classification boundaries see the following subsections.

1.1. Adversarial Framework

Let f be a neural network trained for a supervised learning task on a collection of data (x, y), where $x \in X \subset \mathbb{R}^D$ and $y \in \{1, \ldots, C\}$ denotes a class label. Here, f is one of a parameterized family of differentiable functions that map x to a representation $z = f(x; \theta) \in \mathbb{R}^C$, where z is interpreted as the logit vector for the class probabilities $\hat{y} = \mathrm{softmax}(z)$. During training, a differentiable loss function L is introduced and used to infer f as $\arg\min_\theta L(y, \hat{y})$, generally through some form of stochastic gradient descent (SGD) [15].

Geometrically, we can consider z as an abstract representation of the data x, where the classes $\{z|y = c\}$ form contiguous regions in the representation space [10]. An illustration is given in Figure 1-1. The *classification margin* of an input x is defined as the supremum [41]

$$\sup \left\{ \varepsilon : \left\| x - x' \right\| < \varepsilon \Rightarrow \hat{y} = \hat{y}' \right\},$$

where $\hat{y}' = \mathrm{softmax}(z')$ is the class prediction of $z' = f(x')$ under f. Intuitively, if an input x has a small classification margin, then a small perturbation may push x across a classification margin into a region associated with a different class. In the context of image data, this means that small, imperceptible changes to an image may be sufficient to cause a misclassification.

From this, we see that models with low classification margins are generally associated with susceptibility to adversarial examples. While normal data $z = f(x)$ associated with class y tends to be distributed near a low-dimensional manifold, adversarial samples $z' = f(x')$ with $y' \neq y$ but $\hat{y}' = \hat{y}$ lie in high-dimensional regions just off of this manifold [43, 16]. This has given rise to two general types of attack strategies:

- *White-Box* attacks assume that the adversary has complete knowledge of the model and methods used to train it. This includes access to the internal data of the model, such as its architecture, weights, and gradients. Attacks of this sort generally attempt some form of perturbation $x_{\text{adv}} = x + \varepsilon \nabla_x L(y, \hat{y})$ of an input x along the gradient of the loss in order to induce a misclassification with minimal distortion ε. A canonical example of this type is the Carlini and Wagner optimization attack [6].
- *Black-Box* attacks, on the contrary, make minimal assumptions on model access. In this setting, the model is treated as an oracle, and attackers have a query budget to probe the oracle on any input of interest in order to estimate the classification boundaries of the model, or to train a proxy model on which to perform a white-box attack [35]. Attacks of this type are often distinguished by the particular method they use to optimize queries against the oracle, for examples see [22] and [4].

A complete survey of defensive methods and how to evaluate them is beyond the scope of this paper, but there are excellent resources that exist. We recommend referring to [6] and [5] for in-depth details on evaluating the defense techniques.

1.2. Defense Techniques

An intuitive way to combat such attacks is to simply obfuscate the gradient of f, which in turn prevents direct evaluation of $\nabla_x L$. Defensive distillation [37, 34] and gradient masking [44] are two examples of this approach; some relevant details are given in Appendix A. While such techniques substantially reduced the effectiveness of early attack algorithms, more recent methods can bypass masking by approximating the original gradients of f [3]. As a consequence, gradient obfuscation techniques are now considered largely ineffective for defense [5].

More recent strategies include adversarial training and ensemble methods [44], gradient regularization [41, 45, 19] to increase the classification margins, and Out-of-Distribution (OOD) detection to identify adversarial perturbations to the input [13, 29, 27, 1]. While not foolproof, many of these have attracted interest as the most promising avenues for robustness to adversarial attacks [23]. Jacobian regularization works to increase the effective distance between training data of distinct classes as they are represented by the internal layers of the neural network. This leads to increased robustness to both noisy and adversarial perturbations, as an adversary is forced to make larger perturbations in order to cause a misclassification. On the other hand, OOD detectors are sidechannel models that are trained to learn the distribution of training data, i.e. “what the input data looks like," and send an alert when the data appears to have been modified in some

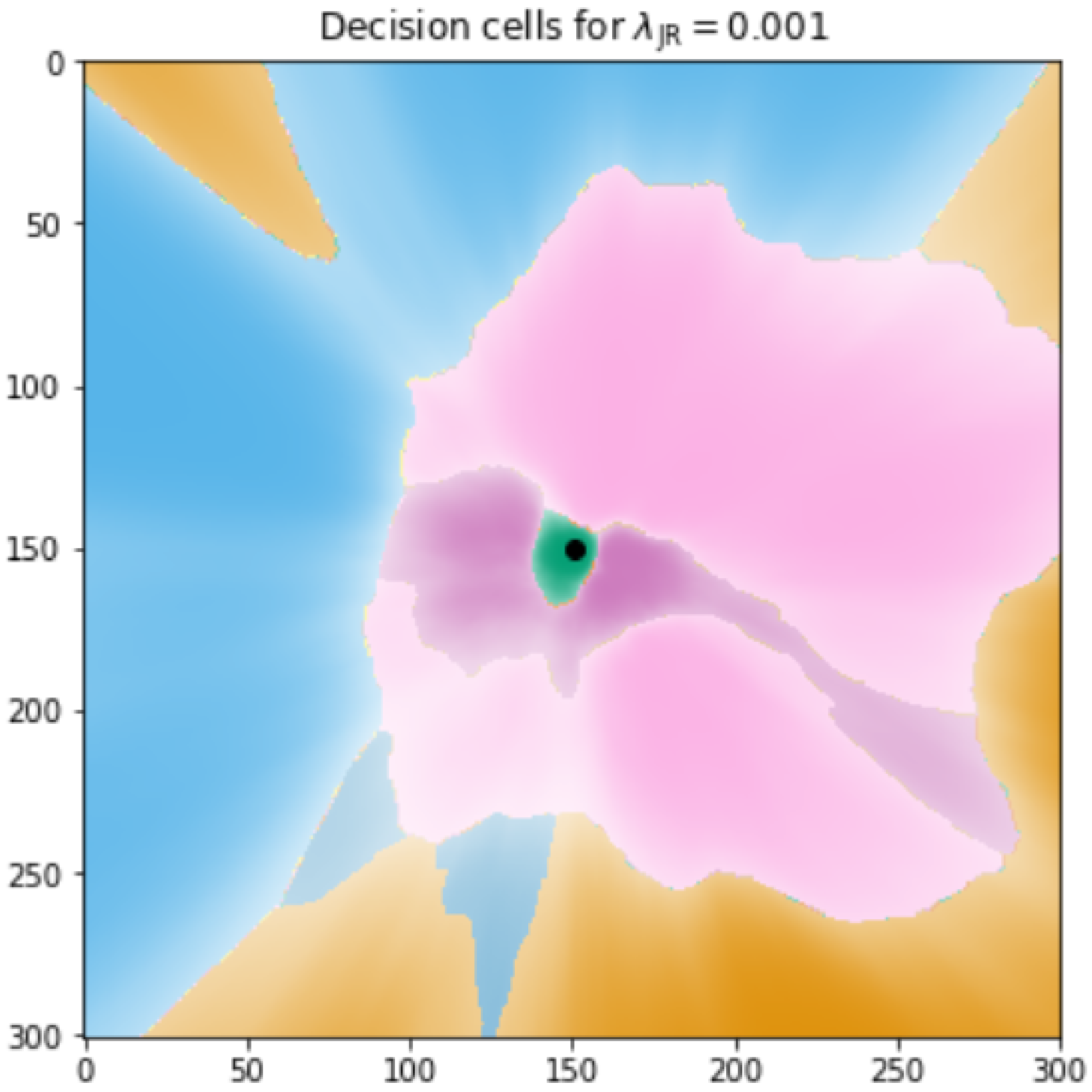

Figure 1-1 Example of classification boundaries for the final layer of a DDNET model [37]. The image represents a random 2D slice of a 3072-dimensional input vector x.

abnormal fashion. It appears that that these two methods would compliment each other well, as forcing the adversary to make larger perturbations would in turn lead to increased detectability. However, to the best of our knowledge, no study has investigated this claim explicitly.

1.3. Our Contributions

Our contribution in this paper is to consider the effect of gradient regularization on the detectability of adversarial samples. In particular, we focus on the Jacobian regularization technique [19], as it can be implemented efficiently in a broad array of neural network architectures. For OOD detection, we consider the Local Intrinsic Dimensionality (LID) technique [29, 20, 21], which separates normal vs. adversarial samples based on their geometric properties, as well as the Deep Mahalanobis (DM) method [27, 23] which builds a generative model for normal vs. adversarial using the representations $z = f(x)$. As with Jacobian regularization, both of these defenses have scalable implementations that make minimal assumptions on the specific structure of the neural network.

As the intent of this paper is to serve as a proof-of-concept, we focus on the readily available CIFAR-10 image dataset benchmark [24], as we can take advantage of a multitude of existing implementations for attacks and associated defenses. As a general rule, we find that adversarial machine learning research in other domains tends to lag behind computer vision, though there has been some success in transferring early attack and defense techniques to audio [9], text [30], and malware detection [11, 42]. These techniques do not yet incorporate the types of defenses we consider here, and we hope that our analysis may serve as a useful first step in evaluating which defensive methodologies to prioritize for transfer to these other fields.

1.4. Threat Model

In order to properly evaluate a proposed defense, we must specify a *threat model* that outlines the goals, capabilities, and knowledge of an attacker. This not only contextualizes the types of threats the defense is intended to mitigate, but also enables us to posit a defensive claim in the form of a falsifiable, and hence scientific, hypothesis [5]. We state at the outset that our goal in this paper is not to provide a comprehensive, rigorous defense to adversarial examples, as this would entail an exhaustive search over a broad range of attack algorithms, associated hyperparameters, and defense-specific variants that is well beyond the scope of this paper (see [7, 3] for some illustrative examples of the difficulty of the defensive problem). Instead, ours is the more modest goal of demonstrating that a combined defensive posture increases the cost of the adversary to find an adversarial example in the form of either lower attack effectiveness or higher rate of detection.

Before stating our claim more formally, we summarize the three major aspects of the threat model as they apply to our work:

- **Adversary Goal**. Given an attack algorithm A, we denote by $x_{\text{adv}} = A(x)$ the adversarial example generated by the attack using clean data x as input. Additionally, let $p = D(x)$ be an OOD detector that maps inputs x to an adversarial probability $p \in [0, 1]$. In our model, the goal of the adversary is to simultaneously minimize the accuracy of $f(x_{\text{adv}})$ on adversarial examples as well as the rate of detection $D(x_{\text{adv}})$ by the sideband detector. We will assume that the adversary intends to conduct both *untargeted* attacks, where x_{adv} is simply classified differently than x, as well as *targeted* attacks, where if x belongs to class c then x_{adv} is to be misclassified a predetermined target $c' \neq c$.

- **Adversary Capabilities**. We assume a white-box scenario where the adversary has complete access to the trained model f, i.e. its architecture, weights, and gradients. Thus we will consider both white-box gradient attacks and black-box label attacks on the model f. We do not place any constraints on the perturbation size $\|x - x_{\text{adv}}\|$ other than implicitly through its relationship to the detectability $D(x_{\text{adv}})$. Additionally, we place no hard constraints on the number of queries the adversary may make against the model, other than computation time for our resources. Finally, we assume that the adversary has access to data x of the same distribution as that used to train f.

- **Adversary Knowledge**. We assume that the adversary is aware that a sideband detector D is in use, but does not know the specific form of that detector. This is a strong assumption, and our motivation comes from the observation than attacks against a specific detector D may involve substantial modifications to be made to the attack (c.f. [3]) that are unable to perform within this scope of work. Instead, we make the weaker assumption that the adversary will perform high-confidence variants of some attacks A to try and defeat the detector in a more generic sense. This is sufficient to evaluate whether our hypothesis of improved cost to the attacker is correct, and we recommend a more thorough defense-aware evaluation in future work.

Let *vacc* denote the mean classification accuracy of model f on adversarial data $A(x), x \in X$, and let *AUROC* be the area under the ROC curve generated by the detector D on a sample of clean (negative) data $D(x), x \in X'$ and adversarial (positive) data $D(x), x \in X$. In light of our threat model, the adversary's goal is to minimize both vacc and AUROC. That Jacobian regularization has a positive effect on vacc was established in earlier work [19]. In this paper, we investigate the following claim:

Hypothesis 1. *Jacobian regularization improves the AUROC performance of an OOD detector D on adversarial examples. Furthermore, this improvement is significant even if its effect on vacc is minimal.*

1.5. Organization

The organization of this paper is as follows. In Section 2, we give an overview of Jacobian regularization. In Section 3, we introduce the LID and DM techniques and provide some details of their implementation. Section 4 gives a thorough discussion of our experimental setup, including regularization parameters, attack techniques, and OOD detector training. Section 5 provides a discussion of our results, and Section 6 concludes.

2. JACOBIAN REGULARIZATION

As discussed in the Introduction, a promising method to defend against attacks is to regularize the network to actively push the classification boundaries farther from the training data, i.e. to introduce large classification margins. In addition to providing robustness to noise, this would have the effect of forcing adversarial perturbations to be larger, and hence more easily detectable. This can be accomplished by adding to the original loss function L a regularization penalty for the Frobenius norm of the model Jacobian $J_{i,j}(x) = \partial z_i / \partial x_j$ [19, 45]. This serves to push the gradient of the model outputs toward zero during the training process, which in turn increases the lower bound on the classification margins [41]. Formally, if we denote by B a batch of training samples (x, y) and λ a regularization strength parameter, the new training loss becomes:

$$L_J(y, \hat{y}) = \frac{1}{|B|} \sum_{(x,y)\in B} L(y, \hat{y}) + \frac{\lambda}{2|B|} \sum_{(x,y)\in B} \|J(x)\|_{\mathrm{F}}^2 . \tag{1}$$

Note that L may include any additional regularization terms, such as traditional L^1 or L^2 weight regularization. The parameter λ controls the influence of this regularization on the overall loss L_J.

As a further illustration, consider the case of a linear model $f(x;\theta) = \theta_1 \cdot x + \theta_0$. Here, Jacobian regularization is equivalent to applying Tikhonov (ridge) regularization to the model parameters θ:

$$L_J(y, \hat{y}) = \frac{1}{|B|} \sum_{(x,y)\in B} L(y, \hat{y}) + \frac{\lambda}{2} \|\theta_1\|_2^2 .$$

From this, we can infer that Jacobian regularization serves a similar purpose to Tikhonov regularization: reducing the model's variance in order to improve stability. In regression settings, this regularization can mitigate the effect of output variable noise (i.e. "outliers") during training. However, for a neural network this also has the effect of flattening out the model's gradient over the entire landscape of the input space, simplifying the geometry of its decision cells, and increasing the classification margins relative to the training data. Not only can this improve the generalizability of the model to input noise, but the shift in the decision geometry forces adversarial perturbations to be larger, and hence confers some measure of robustness to attacks [19].

As with any regularization scheme, Jacobian regularization adds a small amount of bias to the model in order to decrease its variance, thus lowering the model's performance on clean, unadulterated data. While it is possible to introduce Jacobian regularization separately for each layer of f, previous work demonstrates that this imposes too tight of a constraint on the model capacity, resulting in substantially lower performance on clean data [19]. This same paper shows that using just the input-output Jacobian J_{ij} above approximately doubles the minimum classification margin on multiple standard test datasets (MNIST, CIFAR-10, and ImageNet) without a significant loss of accuracy on clean data. They also report improved effectiveness over traditional forms of regularization, such as L^2 weight regularization and dropout.

Automatic differentiation is included with nearly all standard deep learning libraries, and so at first glance implementing Jacobian regularization into the loss (1) appears straightforward.

However, the time required for computing the full Jacobian scales linearly with the number of target classes in the dataset, and this requirement becomes prohibitively expensive for data with a large number of classes. Fortunately, a good approximation to the exact Jacobian norm can be obtained in constant time by first computing a lower-dimensional random projection of the vector-Jacobian product [19]. The approximate Jacobian norm using n_{proj} random projections is:

$$||J(x)||_{\text{F}}^2 \approx \frac{1}{n_{\text{proj}}} \sum_{\mu=1}^{n_{\text{proj}}} \left(\frac{\partial(\hat{v}^{\mu} \cdot z)}{\partial x} \right)^2 \tag{2}$$

where each $\hat{v}^{\mu}$ is a random unit vector. For a large enough batch size $|B|$ consisting of i.i.d. samples, the error of this Jacobian norm estimate is of order $(n_{\text{proj}}|B|)^{-1/2}$ [19]. The computational efficiency obtained through the use of this approximation makes the approach practical even for datasets with a large number of classes, such as ImageNet.

3. OUT-OF-DISTRIBUTION DETECTION

Adversarial detection is often considered as a special case of out-of-distribution (OOD) detection, where the goal is to identify input samples that are drawn from a different distribution than those used to train a model f. In the adversarial setting, the presumption is that adversarial samples $x_{\text{adv}} = A(x)$ generated by some attack A on clean data x have a significantly different distribution than the original data generating process. Here, we discuss two prominent OOD techniques that we selected for this study based on promising results they demonstrated in adversarial settings.

3.1. Local Intrinsic Dimensionality

A common strategy for adversarial detection follows from the intuition that normal samples lie on a low-dimensional manifold in the representation space, while adversarial samples lie in contiguous, high-dimensional regions just off of this manifold [43, 16]. On this view, adversarial samples are identified as lying in low-probability regions of the representation space with local distributions that are significantly different than those of in-distribution data. While these differences can be measured using kernel density estimates [13], more recent and effective techniques take advantage of *intrinsic dimensionality* [20], [21] to distinguish between the high-dimension adversarial regions and low-dimension data regions. Here, we consider the *Local Intrinsic Dimensionality* (LID) approach of [29] that makes use of a computationally efficient estimate for the local dimension of a data sample.

Following [20], LID estimates local dimensionality as the rate of growth of the number of samples in a ball of radius r centered at a test sample x. For example, in Euclidean space the ratio of the rate of growth of the volume of an m-ball to its radius is given by:

$$\frac{V_1}{V_2} = \left(\frac{r_2}{r_1} \right)^m \Rightarrow m = \frac{\ln(V_2/V_1)}{\ln(r_2/r_1)}. \tag{3}$$

LID extends this concept to continuous, real-valued data distributions by using the probability mass in place of volume. Formally, LID is defined as [29]:

Definition 1. *Given a data sample $x \in X$, let $R > 0$ be a random variable denoting the distance from x to other data samples. If the cumulative distribution function $F(r)$ of R is positive and continuously differentiable at $r > 0$, the LID of x is given by:*

$$LID_F(x,r) = \lim_{\varepsilon \to 0} \frac{\ln(F((1+\varepsilon)r)/F(r))}{\ln(1+\varepsilon)} = \frac{r \cdot F'(r)}{F(r)}, \tag{4}$$

$$LID_F(x) = \lim_{r \to 0} LID_F(x,r) \tag{5}$$

whenever the limit exists.

Observe that (4) corresponds to (3) through an application of L'Hôpital's rule. To actually compute LID in practice, [29] make use of a Maximum Likelihood Estimator developed by [2] that estimates (4) in terms of the distances r_i to the k nearest neighbors of x:

$$\mathrm{LID}^*(x) = -\left(\frac{1}{k}\sum_{i=1}^{k} \log \frac{r_i(x)}{r_k(x)}\right)^{-1}. \tag{6}$$

Given a model f and input x, we can apply (6) to the representation $z_l = f(x)_l$ of the l-th layer of the model to define the LID score $L_l(x) = \mathrm{LID}^*(z_l)$ for x at that layer. The detector is then created by training weights α_l from the per-layer LID scores of a collection of adversarial (positive) and clean (negative) samples. In practice, the negative class is generally extended to include samples of clean + white noise data, as the detector should be robust to noisy inputs while still identifying the adversarial ones [12], [13]. The details of this scheme are presented in Algorithm 1. For additional notation, we denote by $r_i(x,S)$ the distance from x to the i-th closest neighbor in set S.

For this work, we use the implementation of [26], which iterates Algorithm 1 over minibatches of size B in order to keep the computation of (6) scalable. The original authors implemented this feature as well [29], and while they report slightly improved results from varying the batch size, we keep it at the default value of $B = 100$ for our experiments. Additionally, this implementation tests the LID classifier over a range of neighbor sizes k on a validation set, and selects the best performing parameter for use on the test set.

3.2. Deep Mahalanobis

Another strategy for OOD detection is to build a generative model for the intermediate representations of in-sample data, generally considered as class-conditional distributions, and identify OOD samples as outliers using various distance measures. The most recent and successful of these methods is the *Deep Mahalanobis* (DM) technique of [27], which models the in-sample representations of each layer of the network as a generative classifier using Linear Discriminant Analysis (LDA). Test samples are scored by their Mahalanobis distance to each mode of the classifier, and OOD samples are identified as those whose maximum score over all classes falls below some threshold.

Algorithm 1: LID training routine

```
Input: Trained network layers $f_l, l \in 1 \dots L$, attack $A$, number of nearest neighbors $k$, and data $X$
1  for $x \in X$ do
2      $X_{\text{noisy}} = \{\text{add noise}(x), x \in X\}$
3      $X_{\text{adv}} = \{A(x), x \in X\}$
4  $N = |X|$
5  LID, LID$_{\text{noisy}}$, LID$_{\text{adv}}$ = zeros$[N, L]$
6  for $l \in 1 \dots L$ do
7      $Z = \{f_l(x), x \in X\}$
8      $Z_{\text{noisy}} = \{f_l(x), x \in X_{\text{noisy}}\}$
9      $Z_{\text{adv}} = \{f_l(x), x \in X_{\text{adv}}\}$
10     for $j \in 1 \dots N$ do
11         $\text{LID}[j,l] = -\left(\frac{1}{k}\sum_{i=1}^{k} \log \frac{r_i(Z[j],Z)}{r_k(Z[j],Z)}\right)^{-1}$
12         LID$_{\text{noisy}}[j,l]$ = repeat line (11) with $Z_{\text{noisy}}$
13         LID$_{\text{adv}}[j,l]$ = repeat line (11) with $Z_{\text{adv}}$
14 $\text{LID}_{\text{neg}} = \text{concatenate}(\text{LID}, \text{LID}_{\text{noisy}})$
15 $\text{LID}_{\text{pos}} = \text{LID}_{\text{adv}}$
16 $\{\alpha_l\} = \text{train classifier}(\text{LID}_{\text{neg}}, \text{LID}_{\text{pos}})$
Return: $\{\alpha_l\}$
```

In detail, let z denote the input to the softmax layer of the network, and $y \in \{1 \dots C\}$ the corresponding class label. Following [27], we use Gaussian Discriminant Analysis (GDA) to model the joint distribution $P(z, y) = P(z|y)P(y)$ as having a Gaussian class-conditional distribution $P(z|y = c) \sim \mathcal{N}(\mu_c, \Sigma_c)$ with unnormalized categorical class prior $P(y) = \beta_c$. From this and Bayes's rule, we can write the posterior class distribution $P(y = c|z)$ as:

$$P(y=c|z) = \frac{P(z|y=c)P(y=c)}{\sum_{c'} P(z|y=c')P(y=c')} \tag{7}$$

$$= \frac{\exp(-\frac{1}{2}z^T\Sigma_c^{-1}z + \mu_c^T\Sigma_c^{-1}z - \frac{1}{2}\mu_c^T\Sigma_c^{-1}\mu_c + \log\beta_c)}{\sum_{c'}\exp(-\frac{1}{2}z^T\Sigma_{c'}^{-1}z + \mu_{c'}^T\Sigma_{c'}^{-1}z - \frac{1}{2}\mu_{c'}^T\Sigma_{c'}^{-1}\mu_{c'} + \log\beta_{c'})}. \tag{8}$$

Now, introducing the LDA assumption that the class-conditional distributions share a tied covariance $\Sigma_c = \Sigma$, this expression simplifies to:

$$P(y=c|z) = \frac{\exp(\mu_c^T\Sigma^{-1}z - \frac{1}{2}\mu_c^T\Sigma^{-1}\mu_c + \log\beta_c)}{\sum_{c'}\exp(\mu_{c'}^T\Sigma^{-1}z - \frac{1}{2}\mu_{c'}^T\Sigma^{-1}\mu_{c'} + \log\beta_{c'})}. \tag{9}$$

Observe that (9) matches the form of a softmax layer with class weights $w_c \to \mu_c^T\Sigma^{-1}$ and biases $b_c \to -\frac{1}{2}\mu_c^T\Sigma^{-1}\mu_c + \log\beta_c$. This suggests the possibility that the representations z were fitted to class-conditional Gaussian distributions in the course of training the network, and that the Mahalanobis distance to the empirical class means can be used as a confidence measure. Indeed, given data (x_i, y_i) and letting $z_i = f(x_i)$ express the softmax layer representations, [27] propose the confidence score:

$$M_c(x) = -(z_i - \hat{\mu}_c)^T \hat{\Sigma}^{-1} (z_i - \hat{\mu}_c), \tag{10}$$

where

$$\hat{\mu}_c = \frac{1}{N_c} \sum_{i:y_i=c} z_i, \qquad \hat{\Sigma} = \frac{1}{N} \sum_c \sum_{i:y_i=c} (z_i - \hat{\mu}_c)(z_i - \hat{\mu}_c)^T \tag{11}$$

are the empirical class means and tied covariance.

On top of this, [27] propose two modifications to improve detection performance. The first is an input preprocessing step whereby the input x is perturbed in the direction of the gradient of the confidence measure:

$$x \mapsto x + \varepsilon \, \mathrm{sign}(\nabla_x M(x)). \tag{12}$$

The motivation for this stems from earlier techniques that used the softmax layer representations directly, where it was observed that the gradients of the softmax for in-distribution data are generally larger than those for out-of-distribution data. As a consequence, the gradient perturbation would more significantly separate the two classes [28]. However, more recent results by [23] suggest that this size difference does not generally hold for the gradients of (10), and for some datasets the perturbation step may in fact decrease performance. The implementation we use accounts for this possibility, as we describe in more detail below.

In a similar manner to LID, the second modification is to extend (10) to all layers of the network, not just the softmax layer, and combine the scores in a feature ensemble. Letting z_l denote the representation of layer l as before, and $\hat{\mu}_{l,c}$, $\hat{\Sigma}_l$ the associated layer statistics, we can write $M_{l,c}(x)$ for the result of (10) applied to this layer. Then, given weights α_l the final confidence score is given by the weighted average $\sum_l \alpha_l M_{l,c}(x)$. As with Algorithm 1, these weights can be inferred by training on a collection of negative (clean, noisy) and positive (adversarial) data.

The DM training procedure with these modifications is presented in Algorithm 2. The overall framework is very similar to that of Algorithm 1, along with the inclusion of clean, noisy, and adversarial samples. For this work, we use the reference implementation [26], which includes a validation step to select the optimal perturbation size parameter ε (including possibly no perturbation, $\varepsilon = 0$). While we omit this validation step in the presentation of Algorithm 2 for brevity, we do include it in our experiments.

The DM authors compare their method in [27] to LID and a number of other contemporary techniques, and obtain very strong results. However, more recent work in [1] and [23] reveals that the tied covariance assumption used to derive (9) is inaccurate in most cases; instead the representations exhibit strong, class-dependent radial correlations. In particular, [23] suggest that in the high-dimensional representation space, adversarial samples frequently have larger variance in directions that are not salient for classification of clean data, and this is what enables tied covariance Gaussians to identify such samples even if they do not properly model the class-dependent distributions themselves. While this observation has led to some iterative improvements to the basic DM algorithm, we do not investigate them here.

```
Algorithm 2: Deep Mahalanobis training routine
Input: Trained network layers $f_l$, attack $A$, perturbation magnitude $\varepsilon$, estimated Gaussian
       parameters $\left\{\hat{\mu}_{l,c}, \hat{\Sigma}_l\right\}$, and input $x$
1  for $x \in X$ do
2      $X_{\text{noisy}} = \{\text{add noise}(x), x \in X\}$
3      $X_{\text{adv}} = \{A(x), x \in X\}$
4  $N = |X|$
5  DM, DM$_{\text{noisy}}$, DM$_{\text{adv}}$ = zeros$[N, L]$
6  for $l \in 1 \ldots L$ do
7      for $j \in 1 \ldots N$ do
8          Find the closest class $\hat{c} = \arg\min_c M_{l,c}(X[j])$
9          Add perturbation $\hat{X}[j] = X[j] + \varepsilon \operatorname{sign}\left(\nabla_x M_{l,\hat{c}}(X[j])\right)$
10         Compute confidence score $DM[j,l] = \min_c \left(M_{l,c}(\hat{X}[j])\right)$
11         $DM_{\text{noisy}}[j,l]$ = repeat lines (8)-(10) with $X_{\text{noisy}}$
12         $DM_{\text{adv}}[j,l]$ = repeat lines (8)-(10) with $X_{\text{adv}}$
13 $\text{DM}_{\text{neg}} = \text{concatenate}(\text{DM}, \text{DM}_{\text{noisy}})$
14 $\text{DM}_{\text{pos}} = \text{DM}_{\text{adv}}$
15 $\{\alpha_l\} = \text{train classifier}(\text{DM}_{\text{neg}}, \text{DM}_{\text{pos}})$
Return: $\{\alpha_l\}$
```

4. EXPERIMENT DESIGN

4.1. Model Baselines and Regularization

In order to better compare our results with those in the existing literature, we constrain our study to model architectures and datasets that are common to the regularization and OOD literature. To that end, this work focuses on convolutional neural networks (CNNs) trained on the CIFAR-10 dataset [24]. Our baseline models consist of:

- DDNET: the Defensive-Distillation network of [37]. This is a relatively simple CNN consisting of two convolutional layers with 64 filters each, followed by a 2×2 max-pooling layer, a second set of two convolutional layers with 128 filters each, another 2×2 max-pooling layer, then a 256-dimensional dense layer followed by a final 10-dimensional dense layer that yields the class logits.
- RESNET34: the 34-layer Residual Network of [17]. This is a large, 34-layer CNN comprising 3.6×10^9 multiply-add operations in its forward pass. We use the implementation of [26] that corresponds to variation B of [17], adapted for use on the CIFAR-10 dataset.

Both models are trained using SGD on the 50,000 CIFAR-10 training dataset, using a batch size of 64, learning rate of 0.1, momentum of 0.9 and an L^2 weight decay of 5×10^{-4}. The DDNET dropout parameter is set to $p = 0.5$, and all model weights are initialized with Xavier

Model	Epochs	Regularization Procedure
DDNET	15	standard regularization
DDNET_T	20	fine-tuned regularization
RESNET	40	standard regularization
RESNET_T	20	fine-tuned regularization

Table 4-1 Model training summary. See [19] and [17] for architecture details of DDNET and RESNET, respectively.

initialization. Jacobian regularization is performed using the implementation library of [18] with λ in the range:

$$\lambda \in \{0, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0\}. \tag{13}$$

We use full projection ($n_{\text{proj}} = 10$) for the regularization (2) as the number of classes in CIFAR-10 is reasonably small.

In addition to this standard regularization procedure, we also consider a "fine-tuning" procedure for each model baseline inspired by transfer-learning techniques [32]. For DDNET, we first train the model using $\lambda = 0$, and then train for an additional number of epochs with $\lambda > 0$ while only re-initializing the final layer. Similarly, for RESNET34 we obtain pre-trained model weights from the Pytorch model zoo [39], and re-initialize just the final layer before continuing training with regularization. The motivation is to gauge the effect of regularization both when it is used in model training from the outset, as well as when it is applied as part of a fine-tuning scheme.

We summarize our baseline models, along with additional training details, in Table 4-1. All training is performed on a single NVIDIA Tesla V100 GPU.

4.2. Adversarial Sample Generation

To evaluate the effectiveness of regularization and OOD detection, we generate adversarial samples using the following attack algorithms. As mentioned in Section 1.3, we do not intend for these to represent a fully comprehensive range of attacks, but enough to encompass our threat model of an attacker with full access to the model f, the intended in-distribution dataset X, and general awareness of a sideband OOD detector D. To that end, we include both gradient-based "white box" attacks and label-based "black box" attacks in our selection, the latter of which makes use of only the classification labels reported by f. We also include "static" attacks that make simple perturbations to the input, as well as more complex "dynamic" attacks that attempt to optimize the perturbation for a given misclassification objective. Our list of attacks includes:

- The Fast Gradient Sign Method (FGSM) [16], which is a simple method that perturbs the input x a fixed distance ε along the gradient of the loss function $L(y, \hat{y})$ associated with the model prediction $\hat{y} = \text{softmax} \circ f(x)$ and the ground truth label y (c.f. Section 2). Formally, the FGSM method can be expressed as:

$$x_{\text{adv}} = x + \varepsilon \, \text{sign}(\nabla_x L(y, \hat{y})). \tag{14}$$

A targeted variant of this method exists whereby the perturbation is adjusted to minimize the loss with respect to a target class t (i.e. $L(y=t,\hat{y})$), but we consider only the untargeted version in this work.

- The Basic Iterative Method (BIM) [25], which is essentially an iterative version of FGSM with an additional clipping step Clip_δ to ensure that the adversarial sample remains in a δ-ball around x. The update at each iteration is given by:

$$x_{\text{adv}}^{k+1} = \text{Clip}_\delta \left(x_{\text{adv}}^{k} + \varepsilon \, \text{sign}(\nabla_x L(y, \hat{y}^k)) \right), \tag{15}$$

  where $x_{\text{adv}}^0 = x$. Again, we consider only the untargeted algorithm here.

- The Carlini and Wagner optimization method (CWL2) [6] is arguably the most effective framework for conducting white-box attacks, as it runs an optimization scheme to maximize a misclassification objective while being constrained to image space, i.e. $x_{\text{adv}} \in [0,1]^n$. To begin, one specifies a differentiable objective function $F(x;t)$ which captures some task-specific measure of confidence that input x will be misclassified as target class t. From this, the optimization problem takes the general form:

$$\arg\min_{x_{\text{adv}}} \|x - x_{\text{adv}}\|_p + c \cdot F(x_{\text{adv}};t), \qquad x_{\text{adv}} \in [0,1]^n. \tag{16}$$

  The optimal x_{adv} is found using gradient descent, and the regularization parameter c is simultaneously fine-tuned using a binary search scheme. The objective function $F(x;t)$ is very general, and a number of examples have been proposed to accommodate various distance measures L^0, L^2, L^∞, untargeted attacks, and even to attack specific defenses [8, 3]. Here, we use their original L^2 objective, which includes an additional confidence parameter κ:

$$F(x;t,\kappa) = \max\left(\max_i \{z_i : i \neq t\}, -\kappa) \right) \tag{17}$$

  where $z_i = f(x)_i$ is the softmax layer input (i.e. logit) for class i under the model f. The parameter κ allows one to target a more confident misclassification at the expense of a potentially larger perturbation $\|x - x_{\text{adv}}\|$, and this is known to have a significant effect on OOD detection schemes [7]. In this work, we consider a targeted low-confidence attack (CWL2_0) with $\kappa = 0$, a targeted high-confidence attack (CWL2_40) with $\kappa = 40$, and an untargeted attack (CWL2_0_U). For a targeted attack on CIFAR-10 input x with true label y, we target the label $t = y + 1 \mod 10$.

- Finally, the Boundary Attack (BA) [4] is a decision-based black box attack that uses no information other than the label returned by the targeted model. Rather than attempt to estimate the gradient, this attack employs an iterative heuristic to estimate the boundary between the true class of x and that of a target class t. While requiring minimal information from the model, there is a tradeoff in that it takes up to $10^4 - 10^5$ queries in order to obtain a sufficiently close adversarial sample. As a result, BA is the most computationally expensive of the attacks that we test, and so we only consider a targeted version. Additionally, it should be noted that BA attacks require a starting point x' for each input x. Thus, for a targeted BA attack on input x, we target the label $t = y + 1 \mod 10$ using a random x' in the current minibatch with true label t. If no such x' is present, we instead target $t = y + 2 \mod 10$ using a corresponding x', and so on as needed.

Attack	Parameters	Comments
FGSM	$\epsilon = 1$	untargeted, Foolbox default
BIM	$\epsilon = .2$, steps = 10	untargeted, Foolbox default
CWL2_0	$\kappa = 0$, steps = 200	targeted, low-confidence
CWL2_40	$\kappa = 40$, steps = 200	targeted, high-confidence
CWL2_0_U	$\kappa = 0$, steps = 200	untargeted, low-confidence
BA	steps = 20000	targeted

Table 4-2 Attack algorithms used to evaluate defenses. BIM uses the default number of steps in [40]. BA and all CWL2 methods use a number of steps estimated from a course binary search on clean data.

We summarize each of these attacks and their relevant parameters in Table 4-2. All attacks are implemented using the Foolbox library [40]. We do not claim that this represents a robust evaluation of hyperparameters, but that it is sufficient to gauge the effectiveness of a combined regularization + OOD detection scheme compared to each defense operating on its own.

4.3. Out-of-Distribution Detection Evaluation

Our scheme for training and evaluating the detectors follows Algorithms 1 and 2. Borrowing the nomenclature of [29], we treat the 50,000 CIFAR-10 training images as a *pre-train* dataset and the 10,000 test images as a *pre-test* dataset. The pre-train data is used to train the baseline and regularized models f as described in Section 4.1. Then, following lines (2)-(3) of both Algorithms, we further subdivide the pre-test dataset into 10% *train* and 90% *test* datasets from which we generate data x_{noisy} with added Gaussian noise and data x_{adv} using the methods in 4.2. The clean data x and noisy perturbations x_{noisy} form the negative class for training the detectors, while the adversarial perturbations form the positive class. Note that we follow the approach of other authors in that we only keep samples x from the pre-test dataset for which both x and x_{noisy} are correctly classified by f. Thus, the size of this and the derivative train/test datasets will depend on the baseline accuracy of the original model, which is reported in our results below.

To train a detector $D \in \{\text{LID}, \text{DM}\}$, we first specify a model f, a validation attack A_{val}, and test attack A_{test}. The validation attack and its corresponding train dataset are used to train the hyperparameters and weights α_l of the detectors, where the clean and noisy samples are considered in-distribution and the adversarial samples out-of-distribution. In the case that a layer l of f outputs a convolutional filter of shape $F \times H \times W$, we follow the procedure of [27] of taking the mean over the H, W dimensions to obtain the vector representation z_l. The detection performance is evaluated against the test attack and corresponding test dataset. As with prior studies, we consider two forms of validation: validating D on the same kind of attack A on which it is to be tested, and validating it on FGSM samples. The latter gives an indication of how well a detector trained on one attack algorithm can generalize to novel, and indeed more sophisticated, algorithms. In our case, this allows us to evaluate whether a detector trained on untargeted FGSM samples can still identify samples from targeted, iterative, and black-box adversarial techniques. Note that even if the detector is validated on the same kind of attack on which it is tested, the validation and testing are performed using different datasets.

To implement Algorithms 1 and 2, we begin with the reference library of [26] which contains code written by the authors of both [29] and [27], and modify the adversarial generation loop to implement the Foolbox library [40] with user-specified attack parameters. We keep their preprocessing steps for normalizing the CIFAR-10 data for the DDNET and RESNET baseline models. As discussed in Section 3, both LID and DM have hyperparameters that must be specified in addition to the layer weights α_l. For LID, this is the nearest-neighbor parameter k, while for DM it is the step size ε. The implementation performs this hyperparameters search during the classifier training step of both algorithms. The hyperparameters are tuned using a grid search on the sets:

$$k \in [10, 20, 30, 40, 50, 60, 70, 80, 90] \tag{18}$$

$$\varepsilon \in [0, .0005, .001, .0014, .002, .005, .01]. \tag{19}$$

The search is performed separately for each $f + A$ combination on the train dataset, and the best parameter from this is used to report results on the test dataset.

Using the pre-train set, we train 32 baseline models f based on the four baseline architectures and eight regularization weights λ of Section 4.1. For each of these, we generate adversarial samples on the pre-test set using one of the four attack algorithms $A \in \{$FGSM, BIM, CWL2_0, BA$\}$. On the 16 transfer models, we additionally generate samples using the CWL2 variants CWL2_40 and CWL2_0_U to measure the influence of high-confidence attacks and untargeted attacks on the detection rate, respectively. This yields a total of 160 f+A combinations on which to train the detectors.

The most time-consuming aspect of this process is generating the adversarial samples for each model f. For CIFAR-10, the pre-train dataset contains 10,000 images. On a single NVIDIA V100 GPU, the FGSM and BIM samples for this dataset can be generated in a few minutes, while the CWL2 samples take approximately 2 hours to process. The BA samples, owing to the large numbers of iterations needed, can take up to 7 hours to process for this dataset.

5. RESULTS AND DISCUSSION

In this section, we evaluate the combined effectiveness of Jacobian regularization and OOD detection to identify adversarial samples. We follow the experimental outline of Section 4, where the model baselines of 4.1 are trained on the CIFAR-10 pre-train dataset and used with the attacks of 4.2 to generate adversarial samples on the pre-test dataset. The DM and LID detectors are trained on the train/test datasets using the evaluation procedure of 4.3.

In view of Hypothesis 1, we expect higher regularization strength to incur an increased cost for the adversary as measured by both improved model accuracy on adversarial samples, i.e. higher *vacc*, as well as increased probability of detection, i.e. higher AUROC. However, this intuitively comes with a concomitant cost to model accuracy on both clean (*acc*) and noisy (*nacc*) samples due to the increased regularization. We will explore the veracity of these statements in the discussion of our results below.

5.1. Effect of Jacobian Regularization on Accuracy

We begin with the effect of Jacobian regularization on the accuracy measures acc, nacc, and vacc. The goal is to corroborate the CIFAR-10 analysis of [19] (Appendix E) and extend it to include a broader range of attacks. Table 5-1 gives the variation clean, noisy, and adversarial sample accuracy with λ for each of the model f + adversarial A combinations, as well as the mean adversarial L^2 perturbation distance. These results are also displayed in Figure 5-1, which shades the 95% quantile region for the adversarial perturbations. We observe two general patterns of behavior that emerge between the optimization-based attacks BA and CWL2 on one hand, vs. the more "static" FGSM and BIM techniques that perform no optimization. First, both BA and CWL2 have a significantly lower vacc (i.e. higher adversarial success rate) across all models and regularizations. In particular, we find that Jacobian regularization has almost no effect on vacc, but it does force the attacks to make larger perturbations in order to maintain this level of effectiveness. We also find that BA and CWL2 demonstrate similar growth in L^2 perturbation distance with λ, which corroborates the assertion that increasing λ pushes the classification boundaries of the model farther away from the training data.

In contrast, the adversarial perturbation of BIM and FGSM are largely unaffected by Jacobian regularization, which follows from the fact that neither of these methods explicitly optimize the perturbation distance. As seen in (14) and (15), both methods perturb the input in a fixed distance based on the sign of the gradient, but do not modify the size of the perturbation. As a consequence, we find that vacc generally increases (i.e. adversarial success rate decreases) with the regularization λ, for rather than attempt to search a larger perturbation distance for an adversarial input, these methods will simply return a failure condition.

These observations indicate that while Jacobian regularization can mitigate the effectiveness of non-optimization attacks, it does little on its own to prevent more sophisticated methods from finding adversarial samples. However, it does force the latter methods to make larger perturbations of the original data in order to cross the classification boundaries. This suggests that gradient regularization should not be used in isolation, but should generally be coupled with OOD detection or some other method to identify perturbations as part of a combined defensive framework.

Concerning the effect of regularization on clean accuracy, Table 5-1 shows that acc generally degrades with increasing regularization strength λ, with a notable exception of the RESNET_T model. In fact, both fine-tuned models DDNET_T and RESNET_T generally exhibit improved accuracy for a given λ over their baseline counterparts. For a possible explanation of this phenomenon, we point to recent work by [14] to study the degree of entanglement of the class-dependent representations of the inner layers of a model and its effect on model performance. They find, rather surprisingly, that *increasing* the entanglement in the internal layers leads to improved generalization accuracy and robustness to perturbations, despite the fact that this would seem to bring the classification boundaries closer to the training data. Their explanation is that the internal layers of the network learn class-agnostic features that are specific to the domain, and it is the final layer that learns discriminatory features for classification. Our results appear to corroborate this, as applying Jacobian regularization to the models from the outset would aggressively separate the class representations across all layers of the network,

	DDNET	$\lambda = 0.000$	$\lambda = 0.001$	$\lambda = 0.005$	$\lambda = 0.010$	$\lambda = 0.050$	$\lambda = 0.100$	$\lambda = 0.500$	$\lambda = 1.000$
	acc	0.793	0.783	0.778	0.772	0.740	0.738	0.628	0.571
	nacc	0.792	0.783	0.778	0.773	0.739	0.737	0.628	0.571
BA	vacc	0.023	0.024	0.027	0.026	0.032	0.031	0.042	0.052
	L^2	1.809	2.245	2.995	3.507	4.871	5.387	8.404	10.752
BIM	vacc	0.348	0.420	0.532	0.568	0.609	0.630	0.588	0.555
	L^2	1.531	1.560	1.579	1.587	1.607	1.614	1.630	1.634
CWL2_0	vacc	0.023	0.024	0.027	0.026	0.032	0.031	0.042	0.052
	L^2	1.390	1.758	2.381	2.776	3.811	4.305	6.726	8.588
FGSM	vacc	0.462	0.507	0.581	0.608	0.627	0.638	0.591	0.554
	L^2	1.660	1.660	1.660	1.660	1.660	1.660	1.660	1.660
	DDNET_T	$\lambda = 0.000$	$\lambda = 0.001$	$\lambda = 0.005$	$\lambda = 0.010$	$\lambda = 0.050$	$\lambda = 0.100$	$\lambda = 0.500$	$\lambda = 1.000$
	acc	0.793	0.786	0.786	0.792	0.756	0.732	0.701	0.644
	nacc	0.793	0.786	0.785	0.791	0.755	0.731	0.701	0.644
BA	vacc	0.023	0.018	0.022	0.024	0.030	0.029	0.029	0.038
	L^2	1.806	2.221	2.955	3.285	4.542	5.428	7.861	9.940
BIM	vacc	0.348	0.430	0.508	0.543	0.616	0.633	0.650	0.618
	L^2	1.531	1.550	1.578	1.583	1.602	1.612	1.626	1.632
CWL2_0	vacc	0.023	0.018	0.022	0.024	0.030	0.029	0.029	0.038
	L^2	1.389	1.757	2.341	2.589	3.607	4.253	6.216	7.986
CWL2_0_U	vacc	0.112	0.118	0.115	0.114	0.129	0.133	0.151	0.167
	L^2	0.824	1.036	1.268	1.365	1.859	2.167	3.013	3.489
CWL2_40	vacc	0.023	0.018	0.023	0.023	0.090	0.100	0.100	0.100
	L^2	7.649	10.792	21.478	35.224	53.823	54.750	54.361	54.318
FGSM	vacc	0.462	0.511	0.557	0.586	0.637	0.644	0.655	0.619
	L^2	1.660	1.660	1.660	1.660	1.660	1.660	1.661	1.660
	RESNET	$\lambda = 0.000$	$\lambda = 0.001$	$\lambda = 0.005$	$\lambda = 0.010$	$\lambda = 0.050$	$\lambda = 0.100$	$\lambda = 0.500$	$\lambda = 1.000$
	acc	0.847	0.831	0.817	0.823	0.837	0.822	0.817	0.807
	nacc	0.845	0.829	0.815	0.822	0.836	0.821	0.815	0.804
BA	vacc	0.016	0.021	0.020	0.017	0.016	0.015	0.024	0.018
	L^2	1.559	1.618	1.851	1.988	2.361	2.674	2.814	2.800
BIM	vacc	0.290	0.306	0.351	0.388	0.464	0.513	0.500	0.527
	L^2	1.543	1.557	1.578	1.589	1.604	1.609	1.610	1.605
CWL2_0	vacc	0.016	0.021	0.019	0.017	0.016	0.015	0.024	0.018
	L^2	1.269	1.313	1.509	1.636	1.933	2.200	2.397	2.969
FGSM	vacc	0.408	0.414	0.449	0.470	0.535	0.566	0.556	0.572
	L^2	1.661	1.661	1.661	1.661	1.661	1.661	1.661	1.661
	RESNET_T	$\lambda = 0.000$	$\lambda = 0.001$	$\lambda = 0.005$	$\lambda = 0.010$	$\lambda = 0.050$	$\lambda = 0.100$	$\lambda = 0.500$	$\lambda = 1.000$
	acc	0.847	0.896	0.899	0.892	0.897	0.903	0.901	0.905
	nacc	0.845	0.891	0.895	0.891	0.894	0.901	0.900	0.903
BA	vacc	0.016	0.011	0.008	0.009	0.009	0.008	0.010	0.008
	L^2	1.561	1.375	1.464	1.508	1.659	1.703	2.040	2.189
BIM	vacc	0.290	0.261	0.313	0.310	0.357	0.384	0.447	0.479
	L^2	1.543	1.502	1.531	1.541	1.552	1.561	1.582	1.585
CWL2_0	vacc	0.016	0.011	0.007	0.009	0.009	0.008	0.009	0.008
	L^2	1.268	1.092	1.207	1.238	1.361	1.392	1.866	1.779
CWL2_0_U	vacc	0.091	0.067	0.062	0.071	0.061	0.061	0.064	0.060
	L^2	0.884	0.828	0.909	0.913	0.993	1.043	1.148	1.221
CWL2_40	vacc	0.098	0.100	0.100	0.100	0.100	0.100	0.100	0.100
	L^2	61.911	66.543	66.664	67.476	66.627	66.246	66.368	66.693
FGSM	vacc	0.408	0.478	0.527	0.511	0.536	0.532	0.559	0.574
	L^2	1.661	1.661	1.661	1.661	1.661	1.661	1.661	1.661

Table 5-1 The variation of clean (acc), noisy (nacc), and adversarial (vacc) accuracy with Jacobian regularization. For each adversarial model, the mean perturbation distance (L^2) is also given.

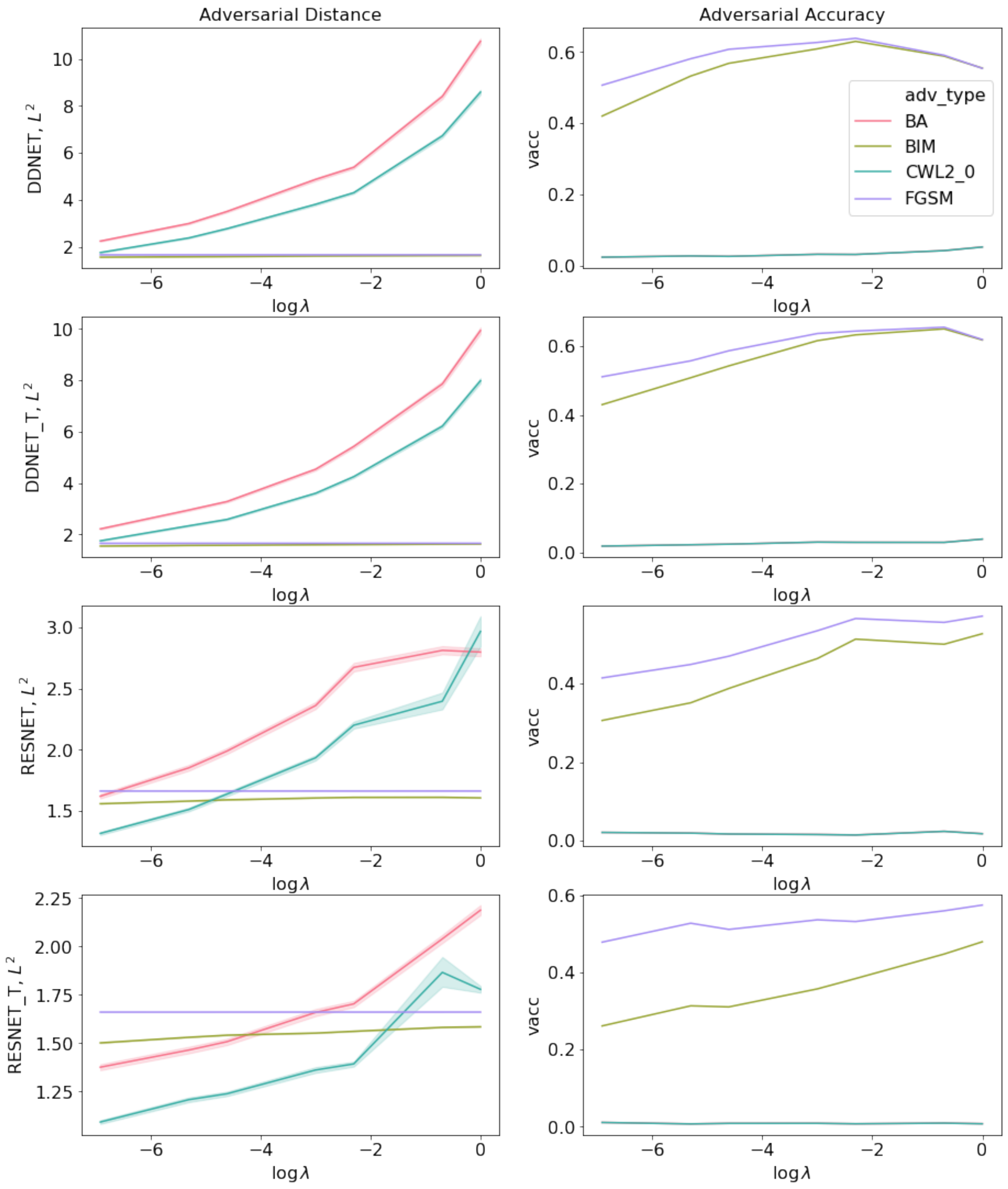

Figure 5-1 Adversarial perturbation distance (L^2) and adversarial accuracy (vacc) vs. Jacobian regularization parameter λ. The shaded regions show 95% quantile regions for distance. The BA and CWL2 accuracies are very similar and appear overlaid on the right-hand plots.

whereas the fine-tuning scheme inherits the near-optimal class boundaries from the pre-trained networks and reinitialized only the final layer.

5.2. Effect of Jacobian Regularization on OOD Detection

Recall from Section 4.3 that the OOD detectors are trained using two general approaches: "self"-trained where both the training and testing are done on samples generated from the same attack, and "other"-trained where the detectors are trained on FGSM samples and tested on samples from the other attacks. This allows us to evaluate the performance of the detectors in the more realistic scenario where they are trained on one form of attack but encounter samples from a novel and previously unobserved algorithm. Table 5-2 gives the performance of the DM and LID detectors trained on FGSM validation samples and applied to each of the other attacks. We measure this performance using the AUROC metric defined in Hypothesis 1. The table gives these results for each $f+A$ combination, along with the corresponding clean model accuracy (acc) as a reference. Results for the self-trained detectors are provided in Appendix B.1.

We begin by comparing our non-regularized OOD results ($\lambda = 0$) with those reported by the original LID [29] and DM [27] authors for the RESNET34 model trained on CIFAR-10. Our detection scores for FGSM, BIM, and CWL2_0 for LID are $(.725, .603, .939)$ respectively compared to the originally reported scores of $(.824, .916, .933)$ in Table 2 of [29] (those authors do not consider BA. Note also that our BIM corresponds to their BIM-b). While our CWL2_0 score is similar, we do not yet know the reason why our scores for the simpler non-optimization attacks are so much lower. While we use a different implementation of the attacks (we apply the Foolbox implementation [40], while they employed the Cleverhans library [33]), we suspect that the discrepancy is more likely due to differences in how the attacks are parameterized, as neither method optimizes its hyperparameters to the same extent that CWL2 does. The vacc scores we report in Table 5-1 for FGSM and BIM are also substantially higher than those given in Table 4 of their paper, which suggests that we have chosen our parameters for these attacks too conservatively. If the perturbations are too minor, then they will neither distinguish themselves significantly from the clean data, nor be as likely to cross a class boundary and result in misclassification. Similarly, our DM detection scores of $(.683, .620, .840)$ are substantially lower than those of $(.999, .989, .939)$ reported in Table 4 of [27]. These are more difficult to compare, since those authors use a custom implementation for these three attacks. It is important to note, however, that as with us neither paper claims to conduct a fully robust suite of attack optimizations. While we recommend a more thorough investigation of attack implementations and hyperparameter choices for future research, this does not prevent us from evaluating the effect of λ on the detection scores which is the primary focus of this work.

Considering the results for $\lambda > 0$, we find that in nearly all cases adding regularization improves detection performance, though both the optimal λ and the magnitude of its effect are case dependent. The improvement appears more significant for the BA and CWL2 attacks, which follows from the sensitivity of their perturbation sizes to λ as seen in Figure 5-1. The effect of regularization is to increase the average distance in representation space from the in-distribution data to the classification boundaries. Thus, as both CWL2 and BA attempt to minimize distance simultaneously with a misclassification objective (i.e. (17) for CWL2 and enforced

	DDNET	$\lambda = 0.000$	$\lambda = 0.001$	$\lambda = 0.005$	$\lambda = 0.010$	$\lambda = 0.050$	$\lambda = 0.100$	$\lambda = 0.500$	$\lambda = 1.000$
	acc	0.793	0.783	0.778	0.772	0.740	0.738	0.628	0.571
DM	BA	0.508	0.598	0.541	0.485	**0.667**	0.561	0.533	0.510
	BIM	0.496	**0.513**	0.500	0.502	0.512	0.505	0.503	0.497
	CWL2_0	0.515	0.612	0.543	0.485	**0.623**	0.551	0.505	0.492
	FGSM	**0.520**	0.515	0.516	0.496	0.515	0.515	0.503	0.505
LID	BA	0.801	0.792	0.795	0.841	**0.842**	0.832	0.736	0.790
	BIM	**0.640**	0.547	0.545	0.548	0.549	0.531	0.521	0.513
	CWL2_0	0.795	0.796	0.797	**0.845**	0.844	0.832	0.740	0.794
	FGSM	**0.651**	0.586	0.562	0.560	0.550	0.533	0.522	0.517

	DDNET_T	$\lambda = 0.000$	$\lambda = 0.001$	$\lambda = 0.005$	$\lambda = 0.010$	$\lambda = 0.050$	$\lambda = 0.100$	$\lambda = 0.500$	$\lambda = 1.000$
	acc	0.793	0.786	0.786	0.792	0.756	0.732	0.701	0.644
DM	BA	0.488	0.573	0.627	0.671	**0.678**	0.570	0.521	0.531
	BIM	0.496	0.505	0.511	0.508	**0.512**	0.496	0.497	0.505
	CWL2_0	0.483	0.571	0.655	**0.674**	0.668	0.546	0.499	0.506
	CWL2_0_U	0.460	0.543	0.570	**0.574**	0.555	0.522	0.512	0.504
	CWL2_40	0.691	0.184	0.060	0.052	0.734	0.392	0.592	**0.999**
	FGSM	0.508	0.504	0.510	**0.525**	0.520	0.495	0.497	0.504
LID	BA	0.781	0.885	0.880	**0.906**	0.814	0.795	0.798	0.739
	BIM	0.465	**0.604**	0.587	0.587	0.545	0.536	0.509	0.506
	CWL2_0	0.789	0.883	0.882	**0.905**	0.815	0.794	0.802	0.746
	CWL2_0_U	0.604	0.809	0.778	**0.818**	0.611	0.606	0.639	0.531
	CWL2_40	0.001	0.941	0.966	**0.995**	0.788	0.785	0.741	0.666
	FGSM	0.560	**0.653**	0.616	0.605	0.550	0.537	0.507	0.510

	RESNET	$\lambda = 0.000$	$\lambda = 0.001$	$\lambda = 0.005$	$\lambda = 0.010$	$\lambda = 0.050$	$\lambda = 0.100$	$\lambda = 0.500$	$\lambda = 1.000$
	acc	0.847	0.831	0.817	0.823	0.837	0.822	0.817	0.807
DM	BA	0.830	**0.840**	0.769	0.775	0.838	0.804	0.775	0.654
	BIM	**0.620**	0.555	0.561	0.557	0.565	0.564	0.549	0.547
	CWL2_0	0.840	**0.853**	0.784	0.786	0.847	0.811	0.785	0.669
	FGSM	0.683	**0.684**	0.668	0.637	0.627	0.623	0.602	0.581
LID	BA	0.938	0.944	0.928	0.926	0.955	**0.960**	0.949	0.906
	BIM	0.605	0.600	0.626	0.617	**0.640**	0.583	0.630	0.618
	CWL2_0	0.939	0.944	0.932	0.926	0.955	**0.959**	0.950	0.910
	FGSM	**0.719**	0.717	0.688	0.667	0.687	0.634	0.667	0.644

	RESNET_T	$\lambda = 0.000$	$\lambda = 0.001$	$\lambda = 0.005$	$\lambda = 0.010$	$\lambda = 0.050$	$\lambda = 0.100$	$\lambda = 0.500$	$\lambda = 1.000$
	acc	0.847	0.896	0.899	0.892	0.897	0.903	0.901	0.905
DM	BA	0.834	0.865	0.843	0.832	0.809	0.858	0.860	**0.867**
	BIM	0.620	0.613	**0.641**	0.613	0.595	0.607	0.606	0.627
	CWL2_0	0.843	**0.890**	0.868	0.853	0.828	0.873	0.868	0.879
	CWL2_0_U	0.767	**0.857**	0.855	0.839	0.820	0.849	0.844	0.827
	CWL2_40	0.128	0.974	**1.000**	0.963	0.869	0.534	0.983	0.167
	FGSM	0.683	0.731	**0.742**	0.712	0.682	0.703	0.672	0.697
LID	BA	0.938	**0.983**	0.947	0.978	0.979	0.971	0.982	0.982
	BIM	0.603	0.642	0.629	0.633	0.626	0.623	0.650	**0.675**
	CWL2_0	0.939	**0.984**	0.946	0.977	0.978	0.971	0.982	0.981
	CWL2_0_U	0.864	0.929	0.798	0.910	0.931	0.934	0.925	**0.940**
	CWL2_40	0.969	0.999	**1.000**	0.990	0.993	0.985	0.997	0.994
	FGSM	0.725	**0.748**	0.734	0.719	0.728	0.732	0.729	0.744

Table 5-2 OOD detection rates with FGSM validation across all models, measured by AUROC. Best scores for each row are highlighted in bold. Clean model accuracies are included for comparison.

misclassification for BA), they must accept larger perturbations from the clean data in order to meet this objective, and thus become more susceptible to detection. A collection of ROC curves for detecting CWL2_0 under FGSM validation is given in Figure 5-2; samples for other attacks are provided in Appendix B.2. The baseline detection rate at $\lambda = 0$ is also significantly higher for the RESNET models than the DDNET ones, and hence the latter tend to show larger gains as regularization increases. For example, adding a moderate amount of regularization ($\lambda = .01$) increases the baseline DDNET_T detection of both BA and CWL2_0 from worse than a coin flip (AUROC $< .5$) to a statistically significant (AUROC $> .67$) at nearly no cost in clean accuracy.

Table 5-2 also demonstrates that adding regularization as a fine-tuning step is generally superior to regularizing the model from scratch, both in terms of clean accuracy and detectability of adversarial samples. This again support the findings of [14] that the internal layers are more strongly associated with general class-agnostic features, and thus regularization should target the discriminatory features of the final layer. Recall that for these models we also consider two additional CWL2 variants: an untargeted low-confidence attack CWL2_0_U and a targeted high-confidence attack CWL2_40. The scores for the untargeted attacks are slightly lower than those for the targeted low-confidence baseline CWL2_0. We believe that this is due to the relaxation of the misclassification criterion in (16), which allows the algorithm to find a tighter bound on the perturbation distance. Far more striking, however, is the degree of improvement for the high-confidence attacks. Observe that the AUROC scores for CWL2_40 are extremely low for $\lambda = 0$ in most cases, but jump significantly as regularization is added ($.691 \rightarrow .999$ for DDNET_T + DM, $.001 \rightarrow .995$ for DDNET_T + LID, and $.128 \rightarrow 1.00$ for RESNET_T + DM). It has been observed (c.f. [7]) that high-confidence attacks can defeat basic OOD detection schemes by forcing the attacker to generate an adversarial sample x_{adv} with a representation z_{adv} that is very similar to those drawn from the target class-dependent distribution. Our results indicate that adding even a small amount of regularization can potentially mitigate the force of such an attack, possibly due to an effect of further separating the class-dependent distributions of the final layers of the model. We recommend investigating this more thoroughly in future work, using adversaries specifically designed to target OOD methods [7, 3].

We also comment on the surprisingly similar behavior of both BA and CWL2 with respect to their adversarial accuracy, perturbation distance, and detectability across our experiments. This is of note because, while both attempt to minimize an objective, CWL2 is a white-box method that requires complete access to the model and its weights, while BA is a wholly black-box approach that only uses the prediction labels. In considering why these very different techniques behave so similarly, we refer to some of the early work in transfer attacks by [36]. In this work, the authors conjecture that, given two similar models f and g and a data distribution D, the representations D_f and D_g of the data under each model will be highly correlated, and hence an attack A that targets the model f may produce samples that are simultaneously misclassified by g. We conjecture that an analogous property holds for samples from attacks A and B that target a single model f. If A and B are both white-box attacks, then it is reasonable to expect that the adversarial samples $A(x)$ and $B(x)$ for a given input x will be correlated, as they will both be perturbations involving the gradient $\nabla_x L(y, P(\hat{y} = y|x))$ at x (c.f. Section 4.2). Would a similar correlation hold if B is instead a black-box attack that does not explicitly use gradient information? We consider this a promising line of investigation for future research.

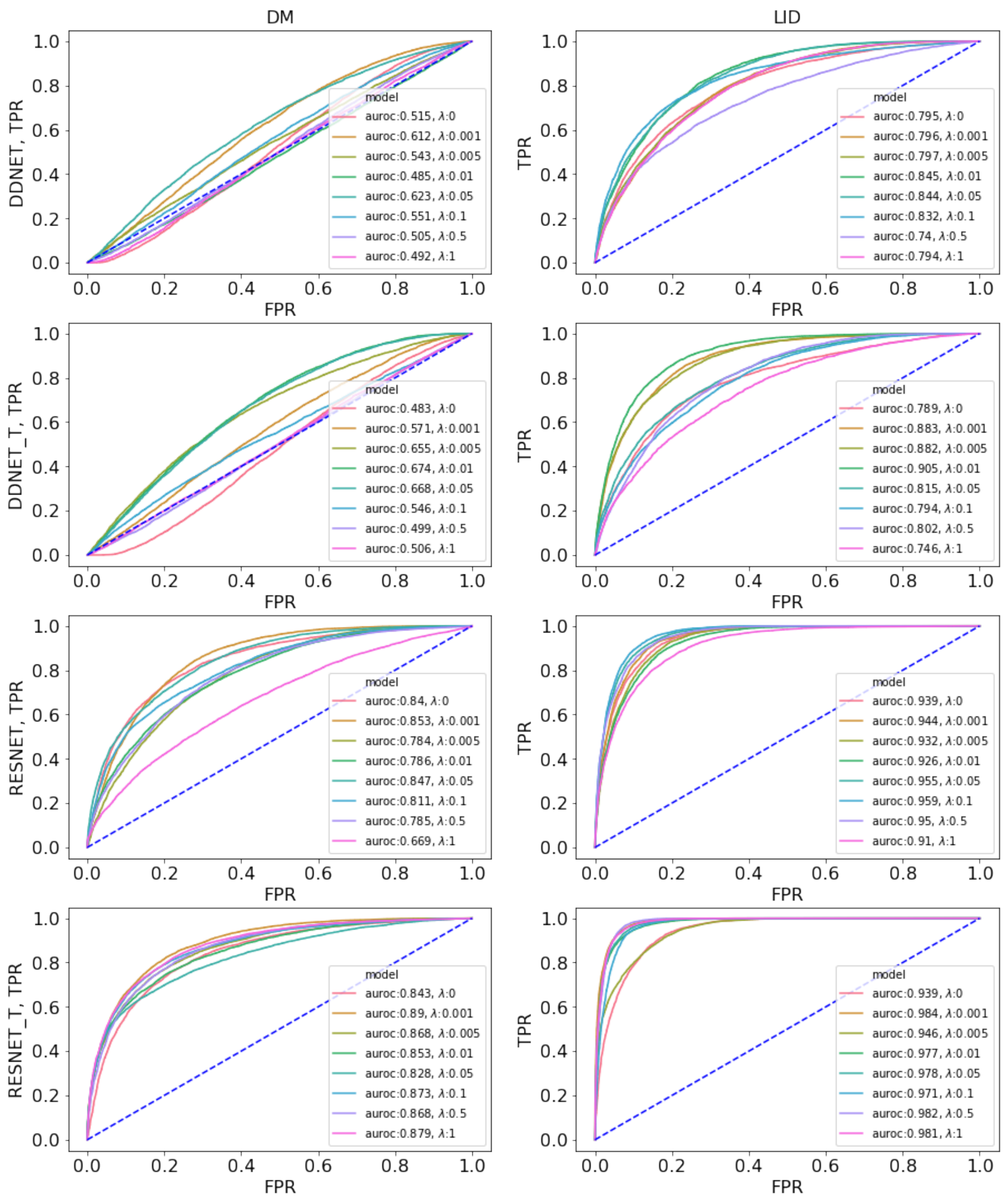

Figure 5-2 ROC curves for OOD detection of CWL2_0 using FGSM validation.

6. CONCLUSION

In this report, we investigated the claim of Hypothesis 1 that gradient regularization techniques improve the performance of OOD detectors to identify adversarial samples. We find that, in most cases, the claim holds, as adding regularization increases the distance from clean in-distribution data to the classification boundaries of the representation layers of the model. As a consequence, either the adversarial accuracy vacc increases (i.e. attacks are less effective), or the adversary is forced to make larger perturbations which increases their detectability. However, the degree of improvement is not uniform, and depends heavily on both the nature of the attack and the accuracy of the baseline model. In particular, we observe that it is easier to identify optimization-based attacks that attempt to minimize the perturbation for a given level of effectiveness than it is to detect static attacks that simply perturb the input for a fixed number of iterations. Additionally, while increased regularization has a more significant effect on adversarial detection for models with a low baseline accuracy(DDNET), the most effective detection performance is obtained for models with a high baseline accuracy (RESNET).

Our results also corroborate the findings of other authors that detectors trained on relatively simple attacks (FGSM) can still effectively identify samples generated from more sophisticated schemes (CWL2, BA). We suggest that this indicates a possible analogy to the well-known phenomenon of transferability of attacks [36], in that the samples x_{adv} generated from a variety of attacks against a single model f may also turn out to be highly correlated. If true, such a property would improve the prospects for detecting previously unobserved attack schemes, and we recommend a thorough investigation of this claim in future work. Additionally, we obtain significantly better detection rates at a lower clean accuracy cost when regularizing pre-trained networks that re-initialize only the final layer. We believe this supports recent claims that deep neural networks learn class-agnostic features in deeper layers and discriminatory features in final layers, which has important implications for understanding which aspects of the network are most salient for improving robustness [14].

We also remark that the adversarial detection rates we obtain are generally lower than those reported by the original LID [29] and DM [27] authors for similar attacks on the CIFAR-10 dataset, even with Jacobian regularization implemented. It is important to note that neither we nor these other studies claim to perform a fully robust battery of adversarial tests, as this involves a level of hyperparameter tuning that is beyond the scope of the current work. To further complicate this search, it is possible to modify certain attacks (such as CWL2) to specifically target adversarial detection schemes [7, 3]. These modifications generally target the confidence score, such as (6) or (10), to produce high-confidence adversarial perturbations. While we are successful identifying adversaries using a generic high-confidence score (CWL2_40), future work should incorporate a threat model where the adversary knows the specific detector in use and explore possible modifications of these attacks.

Our final comments concern the generalization of these techniques to models trained in other data domains, such as text [30] and cybersecurity [11, 38]. An advantage of the regularization and OOD detection methods studied here is that they make minimal assumptions on the neural network architecture, and can be broadly applied to feed-forward, convolutional, recurrent, and even more esoteric models. This makes them amenable to potentially improving the robustness of machine learning tasks that have thus far focused on distillation and model ensembles for security, such as malware classification [42]. Indeed, we conjecture that regularization can be employed to increase the diversity of models within an ensemble, so that an attacker that attempts to attack one model is increasing their likelihood of detection with respect to another, for it becomes increasingly difficult for the adversary to perturb an input in such a way that is simultaneously misclassified by all models and also goes unnoticed by associated sideband detectors. Again, we suggest that this is a promising avenue for future work to extend these defenses beyond computer vision tasks.

REFERENCES

[1] Nilesh A. Ahuja, Ibrahima Ndiour, Trushant Kalyanpur, and Omesh Tickoo. Probabilistic Modeling of Deep Features for Out-of-Distribution and Adversarial Detection. *arXiv e-prints*, page arXiv:1909.11786, September 2019.

[2] Laurent Amsaleg, Oussama Chelly, Teddy Furon, Stéphane Girard, Michael E. Houle, Ken-ichi Kawarabayashi, and Michael Nett. Estimating local intrinsic dimensionality. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 29–38, New York, NY, USA, 2015. Association for Computing Machinery.

[3] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. volume 80 of *Proceedings of Machine Learning Research*, pages 274–283, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[4] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. *arXiv e-prints*, page arXiv:1712.04248, December 2017.

[5] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On Evaluating Adversarial Robustness. *arXiv e-prints*, page arXiv:1902.06705, February 2019.

[6] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. *arXiv e-prints*, page arXiv:1608.04644, August 2016.

[7] Nicholas Carlini and David Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. *arXiv e-prints*, page arXiv:1705.07263, May 2017.

[8] Nicholas Carlini and David Wagner. MagNet and "Efficient Defenses Against Adversarial Attacks" are Not Robust to Adversarial Examples. *arXiv e-prints*, page arXiv:1711.08478, November 2017.

[9] Nicholas Carlini and David Wagner. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. *arXiv e-prints*, page arXiv:1801.01944, January 2018.

[10] Kevin K. Chen. The upper bound on knots in neural networks, 2016.

[11] George Dahl, Jay Stokes, Li Deng, and Dong Yu. Large-scale malware classification using random projections and neural networks. In *Proceedings IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2013.

[12] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. *arXiv e-prints*, page arXiv:1608.08967, August 2016.

[13] Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. Detecting Adversarial Samples from Artifacts. *arXiv e-prints*, page arXiv:1703.00410, March 2017.

[14] Nicholas Frosst, Nicolas Papernot, and Geoffrey Hinton. Analyzing and Improving Representations with the Soft Nearest Neighbor Loss. *arXiv e-prints*, page arXiv:1902.01889, February 2019.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[16] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv e-prints*, page arXiv:1412.6572, December 2014.

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

[18] Judy Hoffman, Daniel A. Roberts, and Sho Yaida. Github - facebookresearch - jacobian regularizer, November 2019.

[19] Judy Hoffman, Daniel A. Roberts, and Sho Yaida. Robust Learning with Jacobian Regularization. *arXiv e-prints*, page arXiv:1908.02729, August 2019.

[20] Michael E. Houle. Local intrinsic dimensionality i: An extreme-value-theoretic foundation for similarity applications. In Christian Beecks, Felix Borutta, Peer Kröger, and Thomas Seidl, editors, *Similarity Search and Applications*, pages 64–79, Cham, 2017. Springer International Publishing.

[21] Michael E. Houle, Erich Schubert, and Arthur Zimek. On the correlation between local intrinsic dimensionality and outlierness. In Stéphane Marchand-Maillet, Yasin N. Silva, and Edgar Chávez, editors, *Similarity Search and Applications*, pages 177–191, Cham, 2018. Springer International Publishing.

[22] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors, 2019.

[23] Ryo Kamoi and Kei Kobayashi. Why is the Mahalanobis Distance Effective for Anomaly Detection? *arXiv e-prints*, page arXiv:2003.00402, February 2020.

[24] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[25] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv e-prints*, page arXiv:1607.02533, July 2016.

[26] Kimin Lee. Github - pokaxpoka - deep mahalanobis detector, August 2019.

[27] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks. *arXiv e-prints*, page arXiv:1807.03888, July 2018.

[28] Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks. *arXiv e-prints*, page arXiv:1706.02690, June 2017.

[29] Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E. Houle, and James Bailey. Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality. *arXiv e-prints*, page arXiv:1801.02613, January 2018.

[30] Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. Adversarial Training Methods for Semi-Supervised Text Classification. *arXiv e-prints*, page arXiv:1605.07725, May 2016.

[31] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. *arXiv e-prints*, page arXiv:1412.1897, December 2014.

[32] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[33] Nicolas Papernot, Ian Goodfellow, Ryan Sheatsley, Reuben Feinman, and Patrick McDaniel. cleverhans v1.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 2016.

[34] Nicolas Papernot and Patrick McDaniel. Extending Defensive Distillation. *arXiv e-prints*, page arXiv:1705.05264, May 2017.

[35] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. *arXiv e-prints*, page arXiv:1605.07277, May 2016.

[36] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks against Machine Learning. *arXiv e-prints*, page arXiv:1602.02697, February 2016.

[37] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks. *arXiv e-prints*, page arXiv:1511.04508, November 2015.

[38] Jugal Parikh, Holly Stewart, and Randy Treit. Protecting the protector: Hardening machine learning defenses against adversarial attacks, August 2018.

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’ Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[40] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017.

[41] J. Sokolić, R. Giryes, G. Sapiro, and M. R. D. Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017.

[42] Jack W. Stokes, De Wang, Mady Marinescu, Marc Marino, and Brian Bussone. Attack and Defense of Dynamic Analysis-Based, Adversarial Neural Malware Classification Models. *arXiv e-prints*, page arXiv:1712.05919, December 2017.

[43] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv e-prints*, page arXiv:1312.6199, December 2013.

[44] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble Adversarial Training: Attacks and Defenses. *arXiv e-prints*, page arXiv:1705.07204, May 2017.

[45] Dániel Varga, Adrián CsiszÃąrik, and Zsolt Zombori. Gradient regularization improves accuracy of discriminative models. 2018.

APPENDIX A. DEFENSIVE DISTILLATION

Here we provide some additional details on defensive distillation [37, 34]. In this process, a secondary model f_T is trained to learn "soft labels" from those predicted by the primary model f. These labels approximate the conditional probability distribution over the outputs given the training data. A temperature parameter, T, regularizes this conditional probability distribution toward a uniform distribution.

In detail, recall from Section 2 that a model $z = f(x)$ is inferred on a dataset (x, y) by minimizing a loss function $\arg\min_\theta L(y, \hat{y})$, where $\hat{y} = \text{softmax}(z)$. To perform distillation, we train a model f_T with the same architecture as f on the dataset $(x, \hat{y})$, i.e. we treat the class predictions of f. Additionally, we replace the standard softmax with the following parameterized version:

$$z_T = f_T(x) \tag{20}$$

$$\hat{y}_{i,T} = \frac{e^{z_{i,T}/T}}{\sum_j e^{z_{j,T}/T}}, \tag{21}$$

where $z_{i,T}$ denotes the i-th component of z_T, etc. The distilled function f_T is thus $\arg\min_\theta L(\hat{y}, \hat{y}_T)$, where the training proceeds as with the original function f. Observe that while $T = 1$ corresponds to the standard softmax function, f_1 is still trained on the dataset $(x, \hat{y})$ as opposed to the original dataset (x, y).

To consider the effect of T, we compute derivatives of $\hat{y}_T$ with respect to the input x to obtain:

$$\frac{\partial \hat{y}_{i,T}}{\partial x_j} = \frac{1}{T}\frac{e^{z_{i,T}/T}}{g^2(x)}\left(\sum_l \left(\frac{\partial z_{i,T}}{\partial x_j} - \frac{\partial z_{l,T}}{\partial x_j}\right) e^{z_{l,T}/T}\right),$$

where g denotes the denominator of (21). From this, we see that increasing T has the effect of reducing the magnitude of the derivatives of $\hat{y}_T$, and in turn reduces the sensitivity of the predictions to perturbations in the input [37]. While this approach proved effective in mitigating many early attack algorithms, it has only a minimal effect on more recent techniques that are able to approximate the gradient of the original network f [6, 3].

APPENDIX B. ADDITIONAL OUT-OF-DISTRIBUTION DETECTION RESULTS

In this Appendix, we give some additional results for OOD detection performance.

B.1. Self-Validated Detector Performance

Table B-1 gives the performance of the OOD detectors when trained on adversarial samples from the same attack on which they are tested, as opposed to Table 5-2 when they are trained only on FGSM samples (note that this means the FGSM results are unchanged, and they are omitted from the table). We find that in most cases, the best AUROC scores are modestly better than the

FGSM-trained values (usually within .05). When comparing to Table 5-2, these results suggest that detectors trained on simpler attacks my generalize to more sophisticated algorithms with comparable performance to those trained on the more sophisticated attacks themselves.

B.2. Gallery of ROC Curves

Here we provide ROC curves for the OOD detection of other attacks. We observe that Deep Mahalanobis largely fails to identify either FGSM or BIM adversarial samples on DDNET, even with the effects of Jacobian regularization. While this may simply reflect the fact that neither attack attempts to optimize perturbation distance, it does contrast with the results of [27] who were successful in identifying such attacks. Similarly, while the LID scores do have discriminatory power with sufficient regularization, they are not as high as those reported by [29]. The Boundary Attack scores have strong discriminatory power across both detector types, and they closely follow the behavior of CWL2 despite the very different nature of the attacks (c.f. Figure 5-2).

	DDNET	$\lambda = 0.000$	$\lambda = 0.001$	$\lambda = 0.005$	$\lambda = 0.010$	$\lambda = 0.050$	$\lambda = 0.100$	$\lambda = 0.500$	$\lambda = 1.000$
	acc	0.793	0.783	0.778	0.772	0.740	0.738	0.628	0.571
DM	BA	0.564	0.622	0.569	0.572	**0.705**	0.651	0.679	0.659
	BIM	0.514	0.500	0.502	0.509	0.517	0.502	**0.528**	0.506
	CWL2_0	0.551	0.634	0.570	0.569	**0.679**	0.612	0.627	0.629
LID	BA	0.806	0.873	0.896	0.894	0.899	**0.902**	0.860	0.872
	BIM	**0.638**	0.619	0.564	0.548	0.547	0.529	0.512	0.517
	CWL2_0	0.811	0.874	0.897	0.896	0.900	**0.903**	0.862	0.874

	DDNET_T	$\lambda = 0.000$	$\lambda = 0.001$	$\lambda = 0.005$	$\lambda = 0.010$	$\lambda = 0.050$	$\lambda = 0.100$	$\lambda = 0.500$	$\lambda = 1.000$
	acc	0.793	0.786	0.786	0.792	0.756	0.732	0.701	0.644
DM	BA	0.569	0.594	0.667	0.705	0.687	**0.714**	0.638	0.586
	BIM	**0.513**	0.498	0.500	0.501	0.511	0.508	0.501	0.501
	CWL2_0	0.577	0.602	0.691	0.699	0.688	**0.710**	0.611	0.561
	CWL2_0_U	0.548	0.559	0.581	0.571	0.551	**0.602**	0.548	0.515
	CWL2_40	0.977	0.996	0.998	0.943	0.996	**1.000**	**1.000**	**1.000**
LID	BA	0.807	0.894	0.903	**0.911**	0.901	0.907	0.849	0.847
	BIM	**0.639**	0.588	0.543	0.562	0.566	0.541	0.509	0.503
	CWL2_0	0.813	0.893	0.905	**0.912**	0.897	0.906	0.856	0.850
	CWL2_0_U	0.793	**0.828**	0.826	0.822	0.782	0.771	0.686	0.616
	CWL2_40	0.999	**1.000**	0.999	0.998	0.963	0.977	0.993	0.996

	RESNET	$\lambda = 0.000$	$\lambda = 0.001$	$\lambda = 0.005$	$\lambda = 0.010$	$\lambda = 0.050$	$\lambda = 0.100$	$\lambda = 0.500$	$\lambda = 1.000$
	acc	0.847	0.831	0.817	0.823	0.837	0.822	0.817	0.807
DM	BA	0.852	**0.853**	0.786	0.805	0.850	0.810	0.780	0.662
	BIM	**0.661**	0.648	0.614	0.587	0.564	0.580	0.550	0.541
	CWL2_0	**0.862**	**0.862**	0.797	0.810	0.854	0.814	0.791	0.667
LID	BA	0.958	0.964	0.950	0.951	**0.967**	0.960	0.949	0.933
	BIM	0.564	0.580	0.612	0.613	0.624	0.572	**0.646**	0.617
	CWL2_0	0.958	0.964	0.950	0.951	**0.966**	0.959	0.949	0.935

	RESNET_T	$\lambda = 0.000$	$\lambda = 0.001$	$\lambda = 0.005$	$\lambda = 0.010$	$\lambda = 0.050$	$\lambda = 0.100$	$\lambda = 0.500$	$\lambda = 1.000$
	acc	0.847	0.896	0.899	0.892	0.897	0.903	0.901	0.905
DM	BA	0.852	**0.925**	0.905	0.875	0.858	0.897	0.884	0.908
	BIM	0.663	**0.756**	0.701	0.686	0.650	0.670	0.624	0.630
	CWL2_0	0.861	**0.929**	0.911	0.883	0.866	0.902	0.886	0.910
	CWL2_0_U	0.823	0.898	**0.902**	0.868	0.857	0.885	0.870	0.891
	CWL2_40	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
LID	BA	0.959	0.983	**0.984**	0.980	0.979	0.983	0.983	0.983
	BIM	0.581	0.602	0.629	0.593	0.616	0.616	0.648	**0.664**
	CWL2_0	0.959	**0.984**	0.982	0.978	0.978	0.981	0.982	0.982
	CWL2_0_U	0.865	0.932	0.939	0.926	0.934	0.938	0.929	**0.947**
	CWL2_40	0.993	**1.000**	0.999	0.999	0.995	0.993	0.998	0.999

Table B-1 OOD detection rates with self-validation across all models, measured by AUROC. The FGSM results are omitted, as they are unchanged from Table 5-2

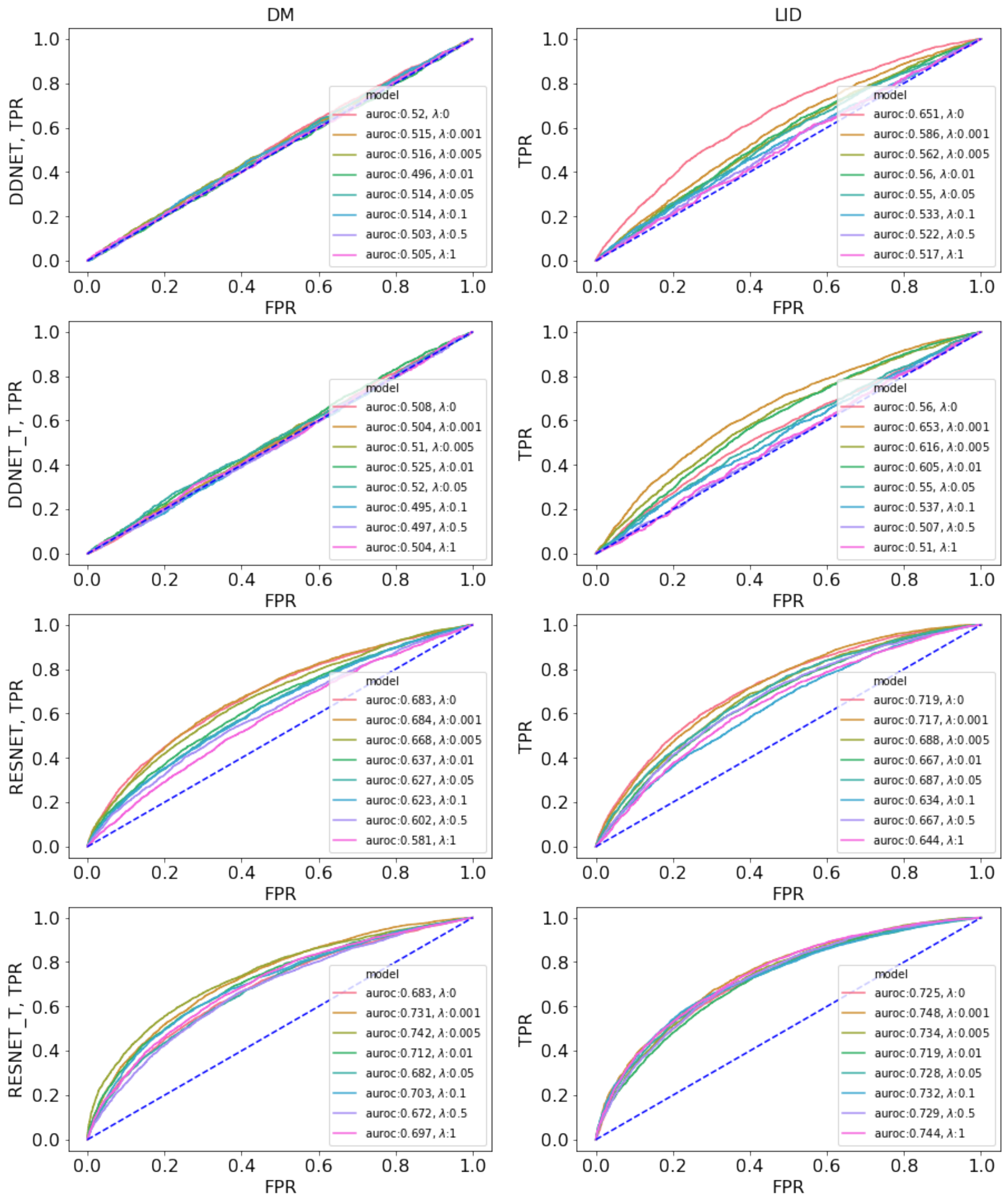

Figure B-1 ROC curves for OOD detection of FGSM using FGSM validation.

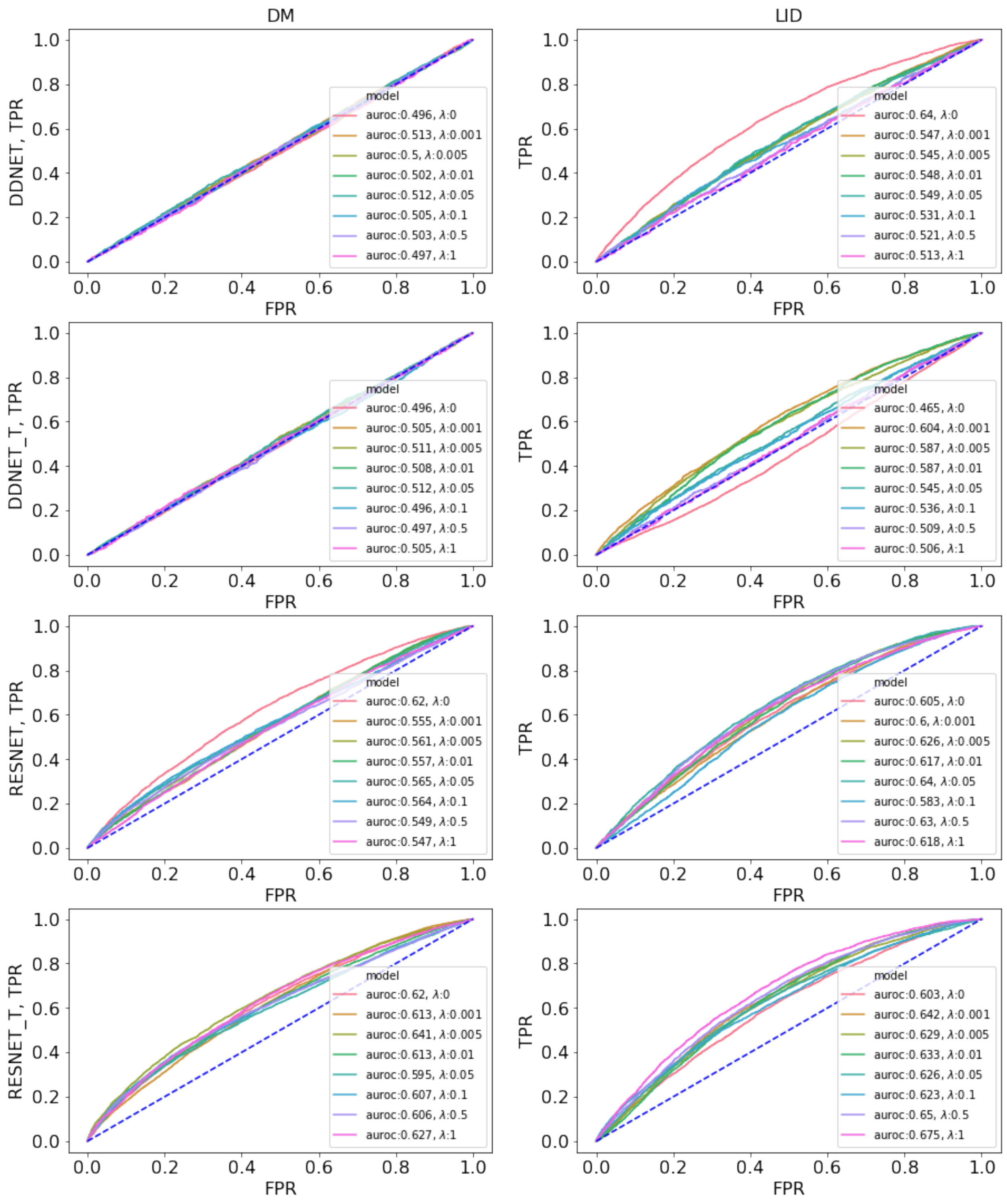

Figure B-2 ROC curves for OOD detection of BIM using FGSM validation.

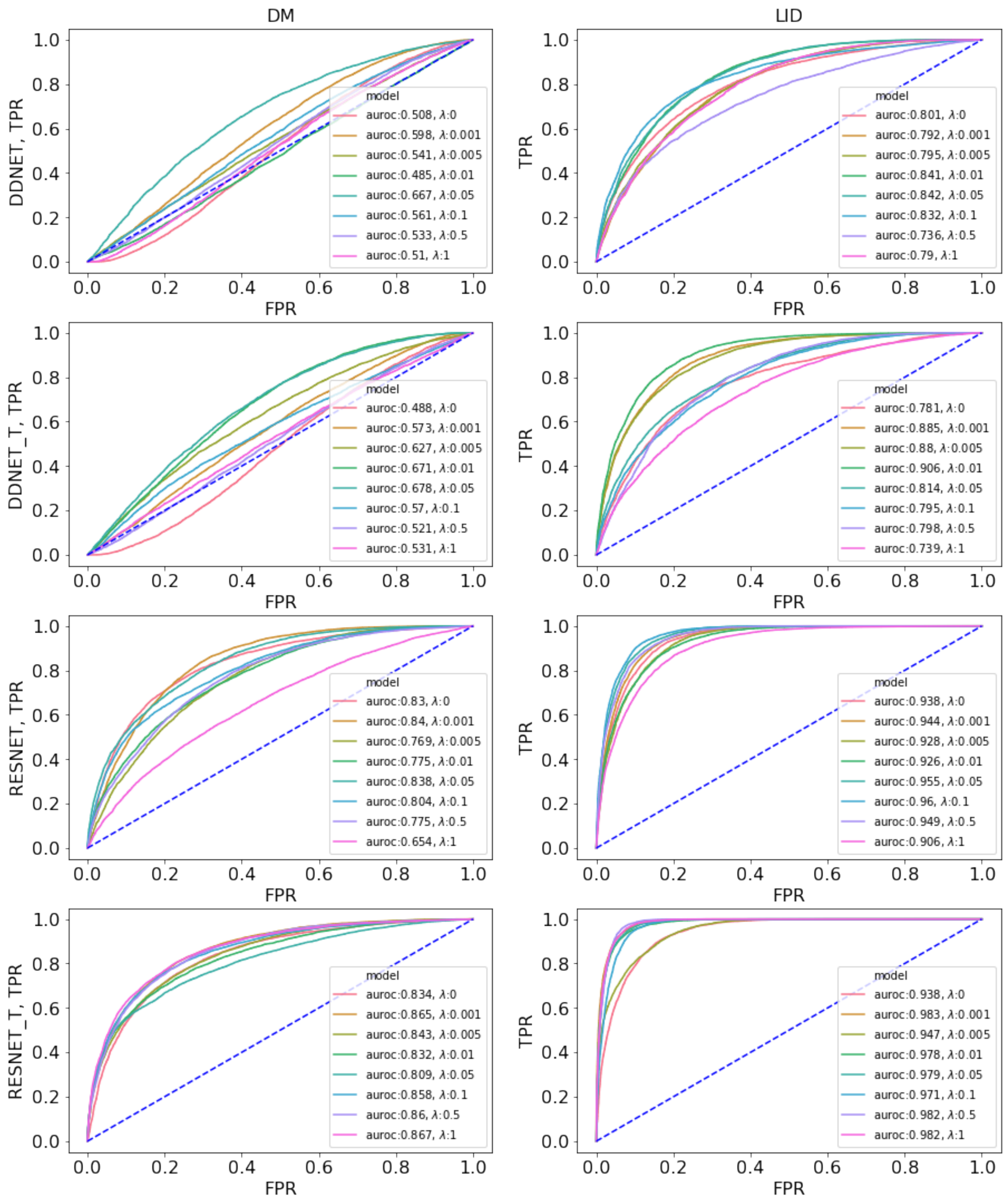

Figure B-3 ROC curves for OOD detection of BA using FGSM validation.

DISTRIBUTION

Email—Internal [redacted]

Name	Org.	Sandia Email Address
Michael Eydenberg	05954	mseyden@sandia.gov
Kanad Khanna	05951	kkhanna@sandia.gov
Ryan Custer	05954	rpcuste@sandia.gov
Daniel Garcia	05951	dgarci2@sandia.gov
Technical Library	01177	libref@sandia.gov

Sandia National Laboratories