

PadNet: Defending Neural Networks Against Adversarial Examples

ARMON BARTON, Computer Science, Naval Postgraduate School, Monterey, United States

MATTHEW WRIGHT, Computer Science, Rochester Institute of Technology, Rochester, United States

SHAIKH AKIB SHAHRIYAR, Computer Science, Rochester Institute of Technology, Rochester, United States

EDGAR JATHO, Computer Science, US Naval Academy, Annapolis, United States

MOHAMMAD SAIDUR RAHMAN, Computer Science, Rochester Institute of Technology, Rochester, United States

KANTHA GIRISH GANGADHARA, Computer Science, Rochester Institute of Technology, Rochester, United States

JIANG MING, Tulane University, New Orleans, United States

Machine learning (ML) suffers from a persistent and critical flaw: adversarial examples. Many new forms of adversarial example attacks have been invented and many narrow defenses have been proposed. Unfortunately, no defensive approach can withstand current attacks. We hypothesize that ML model robustness can be improved with approaches that delineate the data-point-sparse latent space between data-dense regions of a model's classification space as a barrier class. We introduce one such defense, PadNet, that builds a barrier class using a combination of training samples that mix multiple classes together. It leverages this barrier class to separate decision boundaries between benign classes with regions of padding. PadNet then implements a gradient regularization strategy that penalizes gradients in the direction of the barrier class, causing the decision boundary to draw tighter around training samples increasing boundary thickness between classes. We evaluate PadNet against a sampling of the most effective state-of-the-art attacks, demonstrating that it offers significant robustness and reliability compared to current defenses. We also test it against adaptive attacks and find that PadNet remains robust against them.

CCS Concepts: • **Computing methodologies** → **Neural networks**;

Additional Key Words and Phrases: Secure machine learning, adversarial examples, deep learning

This material is based upon work supported by the National Science Foundation under Grants Nos. 2429835, 2422241, and 2040209.

Authors' Contact Information: Armon Barton (corresponding author), Computer Science, Naval Postgraduate School, Monterey, California, United States; e-mail: armon.barton@nps.edu; Matthew Wright, Computer Science, Rochester Institute of Technology, Rochester, New York, United States; e-mail: matthew.wright@rit.edu; Shaikh Akib Shahriyar, Computer Science, Rochester Institute of Technology, Rochester, New York, United States; e-mail: as8751@rit.edu; Edgar Jatho, Computer Science, US Naval Academy, Annapolis, Maryland, United States; e-mail: jatho@usna.edu; Mohammad Saidur Rahman, Computer Science, Rochester Institute of Technology, Rochester, New York, United States; e-mail: saidur.rahman@rit.edu; Kantha Girish Gangadhara, Computer Science, Rochester Institute of Technology, Rochester, New York, United States; e-mail: kantha.gangadhara@rit.edu; Jiang Ming, Tulane University, New Orleans, Louisiana, United States; e-mail: jming@tulane.edu.



This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

© 2026 Copyright held by the owner/author(s).

ACM 2471-2566/2026/03-ART17

<https://doi.org/10.1145/3799889>

ACM Reference Format:

Armon Barton, Matthew Wright, Shaikh Akib Shahriyar, Edgar Jatho, Mohammad Saidur Rahman, Kantha Girish Gangadhara, and Jiang Ming. 2026. PadNet: Defending Neural Networks Against Adversarial Examples. *ACM Trans. Priv. Sec.* 29, 2, Article 17 (March 2026), 26 pages. <https://doi.org/10.1145/3799889>

1 Introduction

Since the initial discovery of adversarial attacks in **Deep Neural Networks (DNNs)** [37], many dozens of these attacks have been invented, and many often narrowly-focused defenses have been proposed in response. Unfortunately, none of the practical defenses proposed have thus far survived counter testing and adaptive attacks [41]. Some have even argued that vulnerability to adversarial examples may be an unavoidable byproduct of how high-dimensional models generalize [36]. At this time, no one effective, computationally acceptable, general approach has been discovered that can shore up DNNs against this and related generalization problems [39, 41].

To address this, we propose a new approach. An ideal defense would identify the data-sparse regions of space between data-dense regions of a classifier’s input [36] and block them off from being used. This would create thick boundary regions between classes such that it would be difficult to find samples that cross between classes with minor modifications. While covering all of the sparse spaces would be impractical, it is only necessary to cover the regions that define the boundaries between classes. Further, we hypothesize that we can define those boundary regions as a new class – the *barrier class* – by generating and training on *padding* – samples related to the original training data.

This training strategy is agnostic of the attack method or the neural network structure. It changes the structure of how the input space is divided by the classifier through training with the barrier class. The approach is modular and able to be added to any DNN classifier, and may augment nearly any present or future defense strategy. The entire approach has a moderate but fixed computational cost – only in training – as opposed to the ever-increasing cost of adding new adversarial attacks for adversarial training.

1.1 Contributions

We introduce *PadNet*, an effective defense against adversarial examples. We demonstrate that PadNet consistently maintains robustness when attacked by the strongest and latest representative set of known attack methods, including **projected gradient descent (PGD)** [23], on which our method scores 0% **attack success rate (ASR)** with 0% false positive rate, compared to 0% ASR with 10% false positive rate for the prior state-of-the-art defense. Additionally, acknowledging valid critiques of the field of adversarial defense by Tramer et al. [41], we develop and evaluate against adaptive attacks to respond to PadNet as a knowledgeable attacker might in the future. Our contributions can be summed up as follows:

- **Boundary Padding:** We introduce *boundary padding (BP)*, which generates interpolated samples between class pairs and assigns them to a dedicated barrier class. This shapes the decision boundary to include a structured region between classes, improving generalization and resistance to boundary-crossing attacks.
- **PGD-Augmented Padding:** We enhance the padding class with adversarial examples generated via PGD, enabling the model to better reject worst-case inputs that attempt to cross into adjacent class regions.
- **Targeted Gradient Regularization (TGR):** We propose a training objective that penalizes gradients pointing toward the padding class, encouraging the classifier to contract its decision regions and reduce vulnerability to directional adversarial attacks.

- **Robustness Across Architectures and Datasets:** We evaluate PadNet on CIFAR-10, CIFAR-100, Tiny ImageNet, and ImageNet-100, including experiments on both CNNs and **Vision Transformers (ViTs)**, demonstrating the approach's robustness across a range of models and data.
- **Boundary Thickness Analysis:** We use *boundary thickness* [47] as a quantitative metric to support PadNet's robustness claims. PadNet achieves significantly thicker decision boundaries than both other defenses and MixUp-based baselines, aligning with its low ASRs.
- **Evaluation Against Adaptive Attacks:** We assess PadNet under multiple threat models, including a specially designed adaptive PGD attack targeting the padding class, a black-box Boundary Attack [4], and a transfer attack. PadNet variants remain effective across these diverse scenarios.

2 Background and Related Work

2.1 Adversarial Example Attacks

Adversarial examples are crafted inputs x' such that given a classifier F , m distinct classes $y_1, y_2, y_3 \dots y_m$, and a benign input x belonging to the class y_x , $F(x') = y_z$ and $y_x \neq y_z$. This is accomplished by adding some small perturbation δ to the input vector x such that $x' = x + \delta$.

An adversarial example x' can be found efficiently through gradient descent and backpropagation. First, let $C(x, y_x)$ be any cost function such that the output is maximal if $F(x) = y_x$ and minimal if $F(x) \neq y_x$. Then, gradient decent can be used to iteratively adjust the pixels in x until the output of C is minimized and $F(x') \neq y_x$. Letting x^t be the value of x' after t iteration steps, we get:

$$x^{t+1} = x^t - \alpha \nabla_{x^t} C(x^t, y_x),$$

where the gradient vector $\nabla_x C(x, y_x)$ is the rate of change of cost function C with respect to changing x to any class other than y_x , and α is a learning rate constant.

2.1.1 PGD Attack. The PGD method is a very effective way to find adversarial examples that are bounded using either the l_∞ or l_2 metrics [23]. PGD is a multi-step variation of FGSM [14] and finds the resulting adversarial example x' by first sampling from a l_∞ -ball around the original example, and then iterating a specified number k times in the direction of the sign of the gradient (for an l_∞ bound) or the direction of the gradient (for an l_2 bound), while not exceeding some specified ϵ .

2.1.2 Carlini–Wagner Attack. The **Carlini–Wagner (CW)** attack [6] is a suite of attacks that use optimization methods to find minimum perturbation according to the l_0 , l_2 , and l_∞ distance metrics. The attack is modeled in terms of an optimization problem as follows:

$$\begin{aligned} & \text{minimize } D(x, x') + c \cdot f(x') \\ & \text{s.t. } x' \in [0, 1]^n \end{aligned}$$

Here, the goal is to find x' such that the distance $D(x - x')$ is minimized. They add the constraint that, for x' to be a valid image, all pixels must be in the range $[0, 1]$. In this problem, the optimization also minimizes $f(x')$, an objective function that outputs a minimal value when $f(x') = y_t$ - i.e., when the attack succeeds. The constant c is used to increase or decrease the weight of f .

2.1.3 DeepFool. Moosavi-Dezfooli, Fawzi and Frossard originally conceived of *DeepFool* as an l_2 -based attack [26], but it can be adapted to any l_p metric. DeepFool uses a greedy algorithm that attempts to approximate the minimum distance to the nearest decision boundary, such that crossing it produces an adversarial example close to the original sample. The authors describe the algorithm as gradient descent using an adaptive step size that it determines at each iteration. The method yields very small perturbations, which the authors believe to be good approximations of the minimum possible perturbation.

2.1.4 Query-Efficient Black-Box Attacks. Recent work has produced a wave of black-box attacks that succeed with far fewer queries than earlier boundary-search methods. Examples include the decision-based *SurFree* attack [24], the score-based sparse *BruSLAttack* [45], and the hard-label bias-synchronised attack of Park et al. [31]. While white-box attacks such as PGD and C&W remain the gold standard for probing worst-case robustness, these newer query-efficient techniques better approximate real-world deployments where gradients are hidden and query budgets are tight. Studying PadNet’s robustness against such attacks will be important future work to understanding its practical security benefit.

2.2 Adversarial Example Defenses

Due to the limited understanding of the nature and extent of vulnerabilities in deep neural nets, defending against adversarial examples has turned out to be a major challenge [7, 29, 41]. Current defensive strategies fall into the following categories: (1) *adversarial training*, (2) *augmentation*, (3) *detecting adversarial examples*, and (4) *external pre-processing*. Approaches (1) and (2) aim to increase robustness of the defended network, while approaches (3) and (4) aim to filter or reject incoming adversarial examples while leaving the defended network unchanged. Currently, the best defenses are a hybrid of two or more of the listed techniques. However, all current defenses fall short in fully securing deep neural nets against adversarial examples.

2.2.1 Augmentation. Several forms of augmentation have been recently proposed to improve model generalization and increase robustness against adversarial examples [15, 17, 28, 34, 48]. Unfortunately, all of these approaches fall short for defending neural networks [16, 41]. Among the major approaches for data augmentation, cutting and mixing-based augmentation methods like CutOut [8], CutMix [48], SaliencyMix [43], ResizeMix [33], and MixUp [50] push the DNN to provide attention to different objects and locations in the input image. All these data augmentation techniques introduce unseen data samples in the training set, which in turn improves the robustness of the DNN models to some degree [28, 43, 48].

Among these, MixUp is the most similar to our augmentation approach, as it introduces linearly interpolation between random pairs of training samples [50]. MixUp augmentation may improve adversarial robustness, because it introduces globally linear behavior in-between the data manifolds [28]. While similar our approach, the key difference is that we ensure our samples exist close to the *decision boundary* and far from other natural samples in the data manifold.

Additionally, different from all of these techniques, our augmented samples are labeled as a “none of the above” class – or simply, the *padding class*. In our evaluation, we show that PadNet far outperforms these augmentation techniques for all attack settings. We also apply these techniques as alternatives to populate the padding class, and find that this leads to much lower benign model accuracy than the PadNet methods.

2.2.2 Detectors. Detectors seek to classify inputs as adversarial examples or benign samples. Working externally to the classifier, they are leveraged in the classification pipeline such that they accept as genuine or reject as adversarial example incoming inputs, passing those accepted on to the classifier. There are numerous detectors in the literature, and a recent comprehensive survey was performed testing 31 of these across four datasets [1].

Based on the results, we choose the four most promising state-of-the-art approaches to adversarial detection for comparison. **Natural scene statistics (NSS)** [19] creates a binary classifier and uses calculated statistical differences between natural images and adversarial examples to differentiate between the two. **Local intrinsic dimensionality (LID)** [22] is another statistically based binary classifier that relies on density differences in the input space between adversarial examples and natural images. **MagNet** [25] uses an autoencoder to remove noise by encoding the image and

decoding the result back into input space. It then uses two metrics within this process to identify adversarial examples. Finally, **neural network invariant checking (NIC)** [21] trains neural networks on the output of each layer of the classifier model for each type of attack sought to be detected, extracting invariants with which to differentiate adversarial attacks.

We demonstrate that our PadNet approaches are more effective than these techniques, with better resilience than all but NIC, and significantly lower false positive rates compared to NIC. NIC also requires attack-specific training that requires knowing the attacks being attempted and scaling the training effort for multiple attacks. PadNet, to the contrary, is attack agnostic and has a training effort that is independent of the attacks being defended against.

2.2.3 Adversarial Training. Currently, the best known method for increasing neural network robustness is Adversarial Training [14, 38]. In this technique, crafted adversarial examples along with their ground truth labels are included within the training set. Unfortunately, Goodfellow et al. showed that the FGSM attack still had a 17.6% success rate against a network defended by adversarial training. Tramèr et al. [42] proposed a more robust method using an ensemble of adversarially trained networks. Kannan et al., however, describe an attack that reduces its black-box accuracy on ImageNet from 66.6% to just 47.1% [18]. Madry et al. propose the state-of-the-art adversarial training technique using PGD examples [23]. However, our experimental results show that Madry et al.'s defense, under many settings, does not provide significantly more robustness compared to an undefended neural network.

2.2.4 TRADES. Zhang et al., TRADES [51], a defense centered around the concept that there is necessarily a tradeoff between natural error (1 - accuracy) and what they describe as robustness error (R_{rob}). Their approach is to minimize a loss function that includes two terms. The first term provides loss between the predicted class and the true label, thereby driving accuracy to increase. The second term uses PGD to encourage the output to be smooth by maximizing the loss provided between a natural example x and a corresponding adversarial example x' divided by a regularization parameter. This has the result of pushing the decision boundary of the classifier away from data samples. TRADES won first place in the NeurIPS 2018 Adversarial Vision Challenge out of over 2,000 submissions. In our TRADES implementation, we found that it has stronger robustness against PGD than what was originally claimed. For white-box attacks against C&W and Deepfool, however, in which the softmax layer was removed from TRADES, we found that robustness was not significantly higher than an undefended neural network.

2.2.5 Gradient Regularization. Gradient Regularization as a concept was first introduced in 1992 by Drucker and Le Cun [11] as a training technique to improve generalization in neural networks, and they called it “double backpropagation.” Ross and Doshi-Valez [35] revive and adapt this approach as an addition to the defensive arsenal against evasion attacks, while also touting its smoothing and generalization benefits. The approach adds a term to the training loss function in the form of the minimization:

$$\theta^* = \arg \min_{\theta} \left\{ \underbrace{H(y, \hat{y})}_{\text{original term}} + \underbrace{\lambda \|\nabla_x H(y, \hat{y})\|_2^2}_{\text{gradient regularization term}} \right\},$$

where H represents the sum of the cross-entropies between the predictions and the labels, and λ is a hyperparameter tuning the regularization term. This causes the model to have smooth input gradients with fewer extreme values. In so doing, the authors found the resulting models to be more resistant to adversarial example evasion attacks. Against PGD, our results show that gradient regularization alone does not provide robustness for higher ϵ values, nor does it provide significantly more robustness compared to an undefended model against stronger attacks such as C&W and Deepfool.

2.2.6 Recent Approaches. Two recent threads refine adversarial training in ways complementary to PadNet. *Augmentation-centric* defenses such as DBAT [20], diffusion-assisted adversarial training [46], and Synth-AT [3] enlarge the training set with synthetic or relabeled examples, whereas PadNet covers the same between-class space with a single padding class and no generative overhead. Meanwhile, *feature-smoothing* methods—including **Adversarial Feature Alignment (AFA)** [32], Inverse Adversaries [9], and GAT [13]—align clean and adversarial embeddings or contract decision boundaries directly. In contrast, PadNet pushes gradients *into* the padding manifold to repel all real classes. These lines of work are orthogonal; combining padding with feature alignment or diffusion-based augmentation is a promising avenue for future gains in robustness.

The recently proposed AFA [32] draws on ideas in both of these threads. It introduces random permuted samples and adversarial examples along with a contrastive loss function applied to feature extraction layers of a DNN, such that the feature maps of input samples become clustered. The clustering minimizes intra-class distance and maximizes inter-class separation for enhanced robustness with limited clean accuracy loss. In contrast, PadNet does not change the feature map representation in embedded space, but instead it envelopes the sparse intra-class and inter-class regions in input space with padding class samples. While AFA improves the robustness accuracy tradeoff, its robustness characteristics are comparable to PGD-AT [23]; we show that PadNet provides much greater robustness than PGD-AT. Moreover, it introduces additional computational overhead due to its reliance on contrastive learning and feature alignment objectives.

GAT [13] similarly includes multiple approaches together: (1) multi-task learning with adversarial training, (2) gradient curvature regularization, and (3) optimization with a customized multi-gradient descent algorithm. The multitask learning adds regularity by augmenting *tasks* as well as data, while PadNet solely augments data with its padding class. The gradient curvature regularization is motivated by the same intuition as PadNet’s **targeted gradient regularization (TGR)**; it guides the model towards flatter loss landscapes which enhances robustness. Although GAT improves adversarial robustness compared to other AT methods, it introduces a loss in clean accuracy by more than 10 percentage points. It also adds substantial training overhead since multi-task learning requires additional loss terms and careful task balancing. Its effectiveness depends on the choice of auxiliary tasks, which may not always align with the primary classification objective and can be difficult to design in practice. Moreover, the added complexity makes training slower and less scalable compared to traditional adversarial training methods.

3 PadNet Design

This section explains how knowledge about adversarial examples informs our design for a new adversarial example defense and describes that design in detail. We focus on two complementary strategies: (1) an augmentation strategy and (2) a gradient regularization strategy.

Unlike Adversarial Training, our strategy does not rely on knowing the methods the attacker uses to create adversarial examples. Therefore, our training set size remains fixed regardless of the number of possible attacks, and our training methods do not change regardless of the attacker’s method.

3.1 Threat Model

We assume an attacker may use an adversarial example x' to perform attacks against a deep neural net F . The attack is successful if and only if the defended network F classifies the adversarial example as a class other than its benign class y or the padding class y_{pad} . In other words, $F(x') \neq y \wedge F(x') \neq y_{pad}$ for a successful attack.

We assume an attacker may perform *black-box* and *white-box* attacks against a deep neural net. A black-box attack is one in which the attacker does not know the parameters θ of F , or the training method. As such, the attacker generates adversarial examples on some state-of-the-art

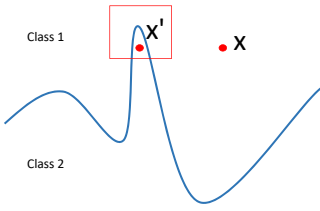


Fig. 1. Data points from within a hyper cube centered around adversarial example x' classify partly as the ground truth label, and partly as other adjacent classes.

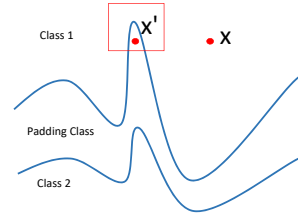


Fig. 2. In boundary padding, a padding class is introduced within the training set in order to reinforce the boundary layer between classes.

deep neural net F' , and subsequently transfers those examples to be classified by the defended network F . A white-box attack is one in which the attacker has full knowledge of the defended network F , and thus knows the parameters θ and the training method. As such, the attacker may generate adversarial examples directly on the defended network F .

3.2 Intuition and Motivation

In supervised learning, the task is to infer a function from labeled training data. Each training example consists of an input feature vector and a desired output value. A supervised learning algorithm analyzes the training data and produces an inferred *decision boundary* that can be used to classify new examples. The placement of this decision boundary is thus very important for the effectiveness of the classifier in generalizing to classify new inputs correctly.

The decision boundary is also important for adversarial examples. Cao et al. [5] performed an experiment in which they formed a small hypercube around the center of an adversarial input x' . They then passed 10,000 data points from within this hypercube through a DNN. This experiment is visualized in Figure 1. They found that a majority of the data points classified as the correct class, while the rest of the data points classified as one or more other classes. Separately, Shafahi et al. [36] provided proofs that adversarial examples necessarily exist within a small ϵ distance of the decision boundary. These empirical and theoretical findings strongly suggest that adversarial examples occur near the decision boundary, and that decision boundaries of multiple classes can exist adjacently to one another.

Using this as motivation, we propose two forms of augmentation that are designed to reinforce the decision boundary against adversarial examples: (1) *BP* and (2) *BP w/ PGD*.

3.3 Boundary Padding

BP is an augmentation strategy that is meant to reinforce the decision boundary between classes. This is done by introducing a *padding class* between the classes in the training data. The model trained on this augmented dataset can be visualized as shown in Figure 2.

Padding class data, or just *padding*, needs to define the regions between classes while not requiring huge amounts of additional data that would overwhelm the training process. We introduce two types of BP samples: (1) *mean padding* and (2) *uniform padding*. In *mean padding*, we take the mean of one random sample x_i from one class and another random sample x_j from another class. Then, we add Gaussian noise to this sample:

$$x_{mp} = 1/2(x_i + x_j) + \mathcal{N}(0, 0.1).$$

Figure 3(c) shows examples of mean padding taken from the two MNIST samples in the top row and the two CIFAR10 samples in the bottom row, respectively. In *uniform padding*, we draw a

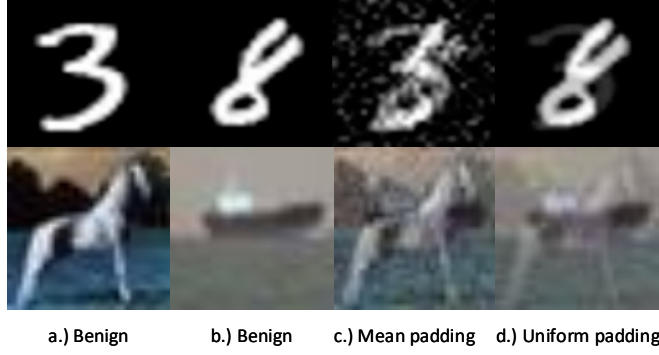


Fig. 3. Mean padding and uniform padding examples for MNIST (top row) and CIFAR10 (bottom row).

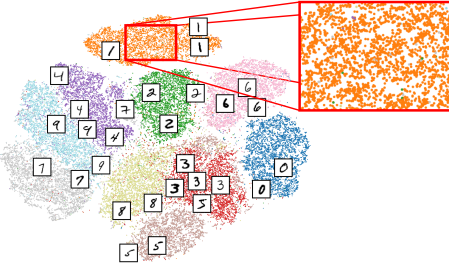


Fig. 4. t-SNE visualization of MNIST classes *before* Boundary Padding or Adversarial Mixup.

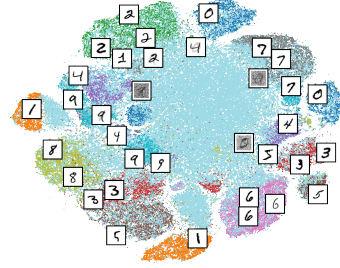


Fig. 5. t-SNE visualization of MNIST classes *after* Boundary Padding.

random variable α from the range $[.2, .8]$. Then we take the weighted average of the two samples using the weights α and $1 - \alpha$, followed by adding Gaussian noise to the sample. For example, setting $\alpha = \mathcal{U} [.2, .8]$,

$$x_{up} = \alpha * x_i + (1 - \alpha) * x_j + \mathcal{N}(0, 0.1).$$

Figure 3(d) shows examples of uniform padding for MNIST and CIFAR10 with $\alpha = 0.2$.

We visualized how BP interacts with the data manifold in 2-D space using t-SNE [44] plots for the original and augmented datasets shown in Figures 4 and 5, respectively. The original dataset in Figure 4 reveals the discrete data-dense regions for each class, with data-sparse regions in between classes. Figure 5 shows the augmented PadNet dataset where the addition of the padding class has altered the data-sparse regions by populating the spaces between classes. As t-SNE is known to preserve local structure between neighboring data points [44], the plots suggest that, in high dimensional space, the padding class reinforces the decision boundary by populating the space in-between data manifolds. This causes the model to draw the decision boundary tighter toward the original classes, removing space where adversarial examples may exist.

While different amounts and combinations of mean padding and uniform padding are possible, we apply a simple approach for our baseline BP method. For each training sample, we added one mean padding sample and one uniform padding sample based on it. For each class (call it class A), we generate padding samples by pairing samples from class A with each other class in equal measure. Padding samples were thus distributed equally across all classes, producing a balanced training set.

3.4 Boundary Padding w/ PGD

We add PGD examples to the augmentation set, along with BP samples, to produce *BP w/ PGD*. Non-targeted adversarial examples were generated using the PGD attack [23], with $\epsilon = 0.25$. Note that, unlike in adversarial training, we do not assume that we can replace PGD with the attacker's actual attack method. In Section 4.4, we show the effect of BP with and without PGD padding against several different attacks. We experimented with different ϵ values and found $\epsilon = 0.25$ to be the most effective. PGD examples were created using 10 iterative steps. This increases the cost of training slightly, though we note that PGD is a relatively fast algorithm and may be used on the fly during training. We show in Section 4.4, however, that it is probably worth this added cost due to further improvements in robustness over the original BP. Additionally, since we only need to construct adversarial examples for a single attack – PGD – we still limit the cost compared to adversarial training against all known attacks.

3.5 Targeted Gradient Regularization

Evidence suggests that adversarial examples occur near the decision boundary [5]. As such, we propose TGR, a technique to regularize gradients in the direction from input x toward the decision boundary. To achieve this, we add a term to the loss function $loss = L(f(x), y, \theta)$ such that

$$loss' = loss + \lambda \frac{1}{n} \sum_{i=1}^n \left(\nabla_x L(f(x), y_{pad}) \right)^2,$$

where $\nabla_x L(f(x), y_{pad})$ is the gradient with respect to x when targeting any sample y_{pad} within the padding class. We penalize high values of the mean of the squared gradient vector. This increases the likelihood that most components in the gradient vector from starting point x toward the *padding class* – and thus toward the *decision boundary* – are near zero. λ controls how much the regularization term contributes to the loss function. In our experiments, we show how increasing λ affects model robustness. Note that changing the loss function means changing the training of the model to create different gradients, not modifying the gradients after the model has been trained. This can impact benign classification decisions, but should not be very harmful when it mostly shifts the gradients away from the decision boundary. We show in our evaluation that the impact on benign classification accuracy is not large.

Since in TGR, the inputs are compared to samples in the padding class y_{pad} , it is important to have a well-formed padding class that truly reinforces the decision boundary. Since we know adversarial examples exist near the decision boundary, we presume that an adversary's goal is to perturb benign samples in a direction *toward the decision boundary* using gradient-based methods. By regularizing this gradient vector during training, TGR shapes the decision boundary to be tighter to the training data in reference to the padding class, and thus to the other classes as well. TGR thus acts to further reduce the input space that adversarial examples can occupy, with a small but non-zero cost to the generalization of the model with respect to benign test samples.

4 Experimental Results

We combine *TGR* with *BP* to construct our approach called *PadNet*. PadNet is a training strategy that is agnostic to any particular dataset or neural network architecture. Hence, the PadNet training algorithm is modular and compatible with other standard deep learning techniques.

To evaluate *PadNet*, we set up experiments in which we trained several variations on MNIST, CIFAR-10, CIFAR-100, Tiny ImageNet, and ImageNet-100. For MNIST, the architecture was a VGG-style CNN as used by Carlini and Wagner [6] and by Papernot et al. [30]. For CIFAR-10, CIFAR-100, and Tiny ImageNet, the architecture was a **Wide Residual Network (WRN)** [49] trained using **Sharpness-Aware Minimization (SAM)** [12]. For ImageNet-100, the architecture was a ViT [10].

Table 1. PadNet Variants

Model	Description
BP	mean and uniform padding only (Isolated)
PadNet	BP + TGR
BP-PGD	mean, uniform, and PGD padding (Isolated)
PadNet-PGD	BP-PGD + TGR

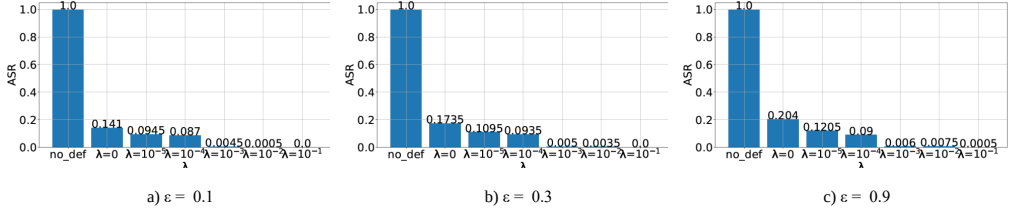


Fig. 6. Attack success rate vs. increasing λ in PadNet's TGR against 2000 PGD adversarial examples with $\epsilon = 0.1$, $\epsilon = 0.3$, $\epsilon = 0.9$ for figures 6a, b, and c respectively.

As shown in Table 1, *BP* indicates that the model was trained with data augmentation via mean and uniform padding only. *PadNet* is BP with TGR applied to the loss function. *BP w/ PGD* is Boundary Padding with the addition of PGD padding. *PadNet w/ PGD* is BP w/ PGD and TGR. The tests without PGD padding allows us to see the impact of just using mean and uniform padding, highlighting their effectiveness and the differences of PadNet from adversarial training.

We measure robustness with ASR, the rate at which a set of adversarial examples cause the neural network to classify a sample x' as any class other than the original, $F(x') \neq y$. For *BP* and *PadNet*, the attack is only considered successful if the adversarial example causes the neural network to classify x' as any class other than the original or the padding class, $F(x') \neq y \wedge F(x') \neq y_{pad}$.

All defenses were evaluated with state-of-the-art attacks that include PGD l_∞ [23], C&W l_∞ [6], C&W l_2 [6], and Deepfool l_∞ [26]. For all attack implementations, we made use of the **Adversarial Robustness Toolbox (ART)** [27].

4.1 Testing TGR

First, we experimented with PadNet's TGR to show the effect of tuning λ , which weights the contribution of TGR to the overall loss function.

The experiment was performed on CIFAR-10 with a PGD white box attack. We trained six PadNet models, each with increasing λ . For the first model, we set $\lambda = 0$, resulting in a model trained with BP augmentation only. For the others, λ was increased from 10^{-5} to 10^{-1} . For each model, we generated 2000 adversarial examples and recorded the ASR. Figure 6(a), (b), and (c) show ASR for PGD with ϵ values of 0.1, 0.3, and 0.9, respectively. The first bar on the x-axis shows 100% ASR for the undefended model. BP alone had ASRs of 14.1%, 17.4%, and 20.4%, respectively. For $\lambda = 10^{-5}$, ASR dropped to 9.4%, 10.9%, and 12%. This trend continues – when λ was eventually increased to 10^{-1} , ASR for all three ϵ values was 0%.

TGR works to flatten *adversarial gradients* within the model. The adversarial gradient is the gradient an adversary uses to find the pixels that are most sensitive towards changing the model prediction for that input. In another words, the adversarial gradient holds key information to inform the adversary what direction to change pixels in order to *ascend* the gradient, thereby increasing loss. As visualizing the loss landscape is intractable, we designed an experiment to indicate that TGR is in fact flattening adversarial gradients. Generally, if the adversarial gradient is flatter, then the

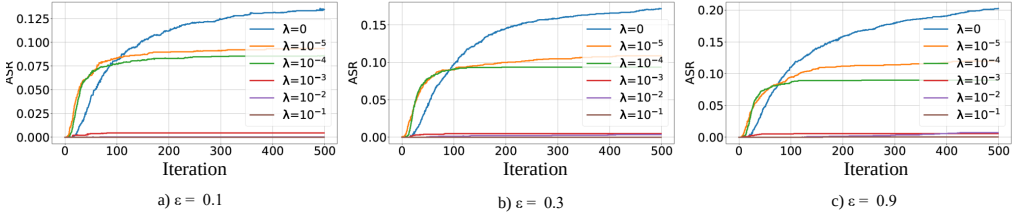


Fig. 7. Attack success rate vs. number of PGD iterations. For each curve, λ is increased from $\lambda = 0.0$ to $\lambda = 0.1$. Increasing λ in PadNet's targeted gradient regularization shows more robustness.

adversary must increase steps taken *up* the gradient to find adversarial examples. As λ is increased, finding adversarial examples should be more challenging.

We generated 2000 PGD adversarial examples using 500 iterative steps and recorded ASR for each step. The results are shown in Figure 7(a), (b), and (c) for ϵ values of 0.1, 0.3, and 0.9, respectively. For each curve, λ was increased from 0 to 10^{-1} . Similar to the previous Figure, $\lambda = 0$ indicates that the model was trained using BP augmentation alone. For $\lambda = 0$, ASR steadily increased as Iterations increased. For $\lambda = 10^{-5}$ and $\lambda = 10^{-4}$, ASR increased to 7 to 10% and then leveled off after approximately 200 iterations. For $\lambda = 10^{-3}$ and above, ASR remained near 0% for all 500 iterative steps. The results suggest that increasing λ in PadNet's TGR improves robustness by flattening adversarial gradients, thus causing adversarial examples to be harder to find.

4.2 PadNet vs. Augmentation Approaches

We have chosen four related augmentation methods to better investigate the distinctions between PadNet and other augmentation strategies: CutOut [8], CutMix [48], SaliencyMix [43], and ResizeMix [33]. Based on our discussion in Section 2.2.1 about these various augmentation methods, we have taken two approaches to evaluate the augmentation methods.

4.2.1 Augmented Samples as Padding. Since BP generates new samples as the combination of samples from different classes, a natural question is whether prior augmentation strategies that combine samples could be used the same way to produce padding. For our first approach, we aim to answer this question by generating samples using each augmentation method and using them to populate the padding class. In Table 2, *CutOut-Padding* indicates that the augmented samples were generated using CutOut and labeled as padding class. These augmented samples were added to the training set and the model structure was modified to accommodate the additional class while training. Similarly, *CutMix-Padding*, *SaliencyMix-Padding*, and *ResizeMix-Padding* models were trained with their respective augmented samples in the padding class. The MNIST and CIFAR-10 column displays the benign accuracy of the corresponding models on those datasets.

As shown in Table 2, all the padding models drastically lose performance when the augmented samples are used as a padding class. This may be surprising given the reported results from those papers. It is important to note, however, that we are using the techniques in a new way to create a padding class, not for traditional augmentation. As such, we can see that techniques with good performance in augmentation are not appropriate for padding. These cutting and mixing strategies generally leave full parts of images intact, such that the label should not be changed to a padding class, but instead kept to their original labels at least in part. Padding samples must instead be selected specifically to be close to the decision boundary to be effective and not confuse the classifier. Our BP strategies, for example, all yield benign classification performance above 97% for MNIST and 87% for CIFAR-10 (see Table 4).

Table 2. Augmentation Models and Their Benign Accuracy on MNIST and CIFAR-10

Model	MNIST	CIFAR-10
Baseline	99.32%	86.77%
CutMix-Padding	70.71%	38.24%
ResizeMix-Padding	45.35%	33.84%
CutOut-Padding	43.10%	31.35%
SaliencyMix-Padding	35.58%	22.65%
CutMix [48]	98.84%	87.58%
ResizeMix [33]	99.21%	88.18%
CutOut [8]	98.68%	86.69%
SaliencyMix [43]	98.99%	87.98%

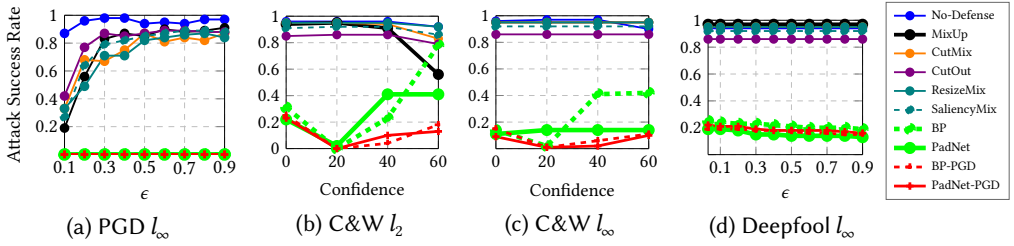


Fig. 8. Augmentation approaches vs. PadNet (CIFAR-10): white-box ASR.

4.2.2 Robustness of the Augmented Models. The second approach considers training the models on MNIST and CIFAR-10 datasets according to CutOut [8], CutMix [48], SaliencyMix [43] and ResizeMix [33] augmentation methods as described in their respective papers. Using state-of-the-art attacks (PGD, C&W, Deepfool), we assess the models' robustness and compare their performance with PadNet. In Table 2, *CutOut* indicates that the models were trained on MNIST and CIFAR-10 where training samples were augmented according to the authors' instructions in [8]. Similarly, *CutMix*, *SaliencyMix*, and *ResizeMix* models were trained according to their respective papers [33, 43, 48]. The benign accuracy of the augmented models in Table 2 suggests that each augmentation technique loses a small amount of benign accuracy on MNIST. Every augmentation method, with the exception of CutOut, somewhat improves benign accuracy for CIFAR-10. PadNet also loses some benign accuracy over the other models as shown in Table 4.

Figure 8 shows the white-box ASR for four attacks against both PadNet variations and augmented models. Against the PGD white-box attack (Figure 8(a)), all the PadNet variants had 0% success rates for all ϵ values. The augmentation approaches all failed against the attack: for $\epsilon \geq 0.5$, ASRs were all 85% and above. We see broadly similar results for Carlini & Wagner l_2 (Figure 8(b)), Carlini & Wagner l_∞ (Figure 8(c)), and Deepfool (Figure 8(d)). The attacker is somewhat more successful against PadNet variants using these attacks. For example, they reach 20%–31% ASRs for Carlini & Wagner l_2 with low confidence, and higher than that by using high-confidence examples against both BP and PadNet without PGD padding. The PGD-based versions of PadNet, however, maintain low ASRs against all attacks. Augmentation approaches, meanwhile, all do poorly against these attacks, with over 80% ASR in nearly all cases.

These findings show that augmentation methods can be effective in enhancing generalization performance, but they are not effective at enhancing robustness. In contrast, the PadNet variants all offer significant robustness. We also see that, when compared to BP, PadNet often significantly increases robustness, emphasizing the value of TGR in addition to BP augmentation.

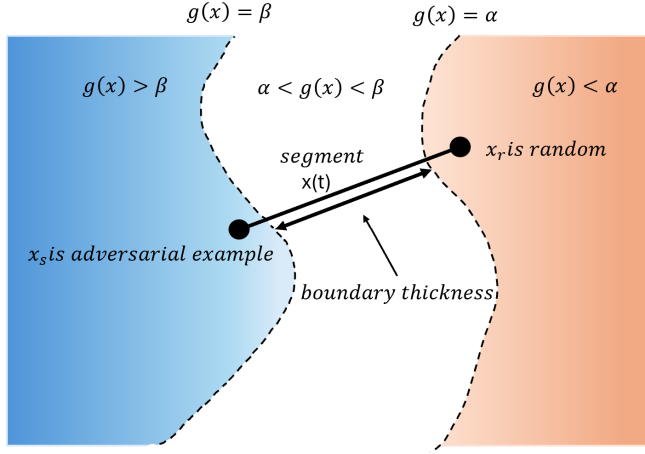


Fig. 9. Intuition behind boundary thickness [47]. Boundary thickness represents the gap between two level sets $g(x) = \alpha$ and $g(x) = \beta$ by measuring the expected distance on random line segments between clean samples in $\mathcal{X}$ and their adversarial example counterparts.

4.2.3 Boundary Thickness. Yang et al. [47] introduced a novel concept called *boundary thickness* to help explain and predict a deep learning model's robustness against adversarial examples. Boundary thickness is defined as the minimum distance one must perturb a data point before the model's prediction changes, averaged over directions in the input space. Intuitively, it measures how “thick” the decision boundary is around a point – thicker boundaries suggest that more substantial perturbations are required to change the model's prediction, indicating greater robustness. Theoretically, the authors show that when a classifier's decision boundary has high curvature or lies close to the data manifold, small perturbations can more easily cause misclassifications. Conversely, flatter, thicker boundaries provide a buffer against such attacks.

Formally, let $f : \mathcal{X} \rightarrow \mathbb{R}^C$ be a classifier that maps inputs $\mathcal{X} \in \mathbb{R}^d$ to class logits for C classes. For neural networks, the function $f(x)$ is the output of the softmax layer, and $f(x)_i$ for class $i \in [C]$ represents the posterior probability $Pr(y = i|x)$, where (x, y) is a feature vector and response pair. The thickness of the decision boundary is then quantified by measuring the posterior probability difference $g_{ij} = f(x)_i - f(x)_j$ on line segments connecting pairs of points $(x_r, x_s) \in \mathcal{X}$, where x_r is chosen uniformly at random from the training set with predicted label i , and x_s is an ℓ_2 adversarial example generated by attacking x_r to a random target class $j \neq i$. Then, the boundary thickness is defined as follows.

Definition 1. (Boundary Thickness [47]) For $\alpha, \beta \in (-1, 1)$ and distribution p over pairs of points $(x_r, x_s) \sim p$, let the predicted labels of x_r and x_s be i and j , respectively. Then, the boundary thickness θ of a prediction function $f(\cdot)$ is

$$\theta(f, \alpha, \beta, p) = \mathbb{E}_{(x_r, x_s) \sim p} \left[\int_0^1 \mathbf{I}\{\alpha < g_{ij}(x(t)) < \beta\} dt \right], \quad (1)$$

where $g_{ij} = f(x)_i - f(x)_j$, $\mathbf{I}\{\cdot\}$ is the indicator function, and $x(t) = tx_r + (1-t)x_s$, $t \in [0, 1]$.

As shown in Figure 9, boundary thickness represents the gap between two level sets $g_{ij}(x) = \alpha$ and $g_{ij}(x) = \beta$ by measuring the expected distance on random line segments between clean samples in $\mathcal{X}$ and their adversarial example counterparts. Figure 10 shows a CDF of empirical boundary thickness measurements for PadNet variants compared to the MixUp techniques for 1000 random

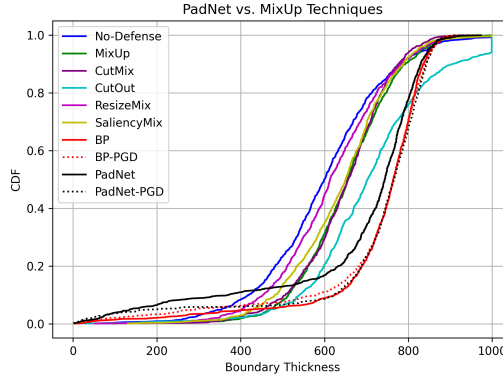


Fig. 10. Probability of boundary thickness of PadNet decision boundaries compared to MixUp techniques tested over 1000 random samples and their adversarial example counterpart.

line segments where t was discretely incremented from 0 to 1 with a step size of 0.001. All PadNet variants performed better than the MixUp techniques. The best performing models were PadNet-PGD, BP-PGD, and BP which all had a 14% increase in boundary thickness in the median case compared to MixUp, CutMix, and SaliencyMix, and a 3% increase compared to CutOut.

The results indicate that PadNet achieves higher robustness and boundary thickness by explicitly shaping its decision boundaries to resist adversarial perturbations. Unlike standard mixup techniques, PadNet introduces mean padding, uniform padding, and adversarial examples (e.g., PGD-perturbed inputs) as part of a dedicated “**none-of-the-above**” (NOTA) class, effectively widening the margin between known classes and out-of-distribution or adversarial regions. This forces the model to reserve high-confidence predictions only for regions well inside class boundaries. Additionally, PadNet employs TGR, which identifies adversarial directions via gradient ascent and penalizes the ℓ_2 norm of these adversarial perturbations in the training loss. This encourages the model to resist label-changing perturbations by smoothing the loss landscape in vulnerable directions. Together, these mechanisms create a “buffer zone” around each class, leading to thicker decision boundaries. As a result, PadNet exhibits greater boundary thickness, meaning adversarial inputs must traverse a larger region before crossing into a different decision region—an indicator of improved local robustness.

4.3 PadNet vs. Other Detectors

We compared BP, PadNet (PN), BP-PGD, and PN-PGD to state-of-the-art adversarial example detection methods recently tested by Aldahdooh *et al.* [1]. In that study, the authors selected eight detectors to evaluate from a field of over 30 competitors and tested them against a barrage of adversarial example evasion attacks over a number of datasets. In Table 3, we select the highest performing detectors in the Aldahdooh survey—NIC [21], LID [22], NSS [19], and MagNet [25], across three datasets—MNIST, CIFAR-10, and Tiny Imagenet, and compare their performance against PadNet variants with respect to attack success and false positive rates. In the table, we rank performance with colors based on the lowest ASR first. Note that significant digits not shown in the table were used to break ties. When ASR is precisely tied, the one with the lower false positive rate is chosen.

With respect to MNIST, the top performing detector was NIC which was first against CW L_∞ and DeepFool, and NSS which was first against PGD. PadNet variants consistently came in second and third place against all three attacks with ASR ranging from 0 to 17, and FPR ranging from 0

Table 3. PadNet Variants vs. State-of-the-Art Detectors Across Three Datasets (MNIST, CIFAR-10, Tiny-ImageNet)

Dataset	Attack	NIC		LID		NSS		MagNet		BP		PN		BP-PGD		PN-PGD	
		ASR↓	FPR↓	ASR	FPR	ASR	FPR	ASR	FPR	ASR	FPR	ASR	FPR	ASR	FPR	ASR	FPR
MNIST	PGD L_∞	0	10	21	1	0	0	0	0	0	0	0	0	0	0	0	1
	CW L_∞	0	10	36	4	98	0	59	0	24	0	17	0	24	0	7	1
	DF	0	10	7	0	2	0	3	0	10	0	3	0	3	0	3	1
CIFAR-10	PGD L_∞	0	10	1	0	7	7	34	1	2	0	0	0	0	1	0	0
	CW L_∞	38	10	35	21	73	7	87	1	14	0	11	0	15	1	9	0
	DF	15	10	36	6	50	7	43	1	12	0	19	0	19	1	11	0
Tiny Imagenet	PGD L_∞	0	10	2	1	16	22	99	1	2	0	1	0	0	0	3	0
	CW L_∞	0	10	100	0	68	22	92	1	40	0	87	0	3	0	2	0
	DF	0	10	69	18	73	23	99	1	88	0	88	0	52	0	51	0

All values are percentages; lower is better. Cells are shaded by rank within each attack–dataset pair (green = 1st, blue = 2nd, orange = 3rd).

Table 4. Defensive Models and Their Benign Accuracy on MNIST and CIFAR-10

Model	MNIST	CIFAR-10
No-Defense	99.52%	92.10%
MixUp [17]	99.43%	92.38%
Madry et. al [23]	99.39%	90.86%
TRADES [51]	97.91%	85.57%
Grad. Reg. [35] A	98.99%	91.88%
Grad. Reg. [35] B	99.36%	90.27%
BP [Ours]	98.15%	90.56%
PadNet [Ours]	98.53%	87.36%
BP-PGD [Ours]	98.40%	89.67%
PadNet-PGD [Ours]	97.80%	89.13%

to 1. For CIFAR-10, PadNet variants outperformed state-of-the-art detectors with PN achieving first against PGD, and PN-PGD achieving first against CW L_∞ and DeepFool. For Tiny Imagenet, NIC scored the lowest ASR in two of three attacks, CW L_∞ and DeepFool, while BP-PGD scored the lowest ASR and FPR against the PGD attack. Additionally, the NIC detector’s FPR of 10% is far behind that of the PadNet variants with 0–2% FPR.

We also note that PadNet offers better deployment features than NIC. Whereas PadNet and its variants do not require specific training for each new evasion attack to enable successful detection, NIC requires training on a full dataset of each attack type’s examples to achieve the performance observed in Table 3. NIC thus suffers much the same drawback as adversarial training, in that each new attack requires a new set of training specific to it.

4.4 PadNet vs. Gradient Regularization

We compared BP and PadNet robustness to an undefended model, and to three other state-of-the-art defenses: Adversarial Training using PGD examples [23], TRADES [51], Gradient Regularization [35] (Grad. Reg. A), and Gradient Regularization combined with Adversarial Training [35] (Grad. Reg. B). The benign accuracy for the models are shown in Table 4 for MNIST and CIFAR-10. PadNet

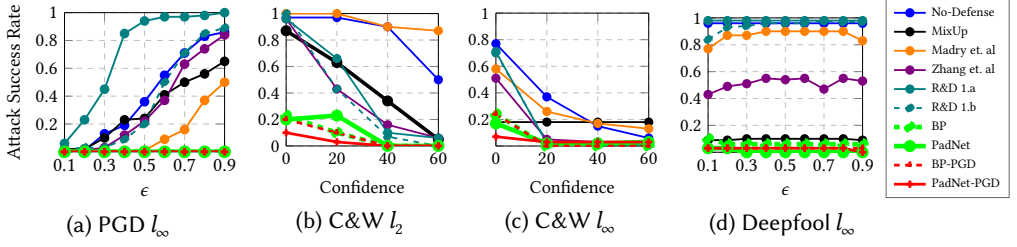


Fig. 11. SOTA Defenses vs. PadNet (MNIST): white-box ASR.

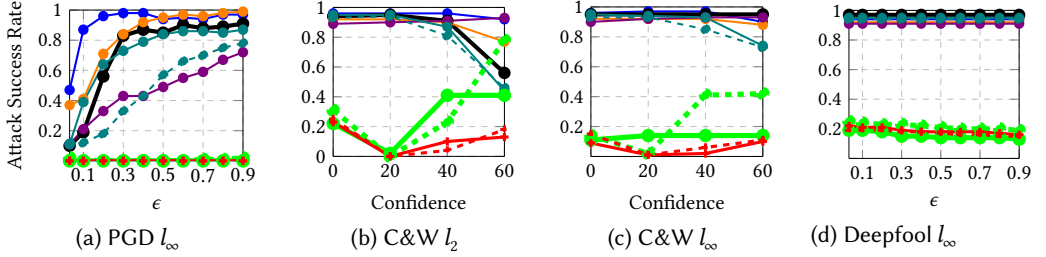


Fig. 12. SOTA Defenses vs. PadNet (CIFAR-10): white-box ASR.

is better than TRADES [51] but loses a bit more benign accuracy than some other models. We will show, however, that it is dramatically more robust than all of these models.

Figures 11(a) and 12(a) show ASRs with increasing ϵ on the x-axis for all models against the PGD white-box attack on MNIST and CIFAR-10, respectively. For low ϵ values on CIFAR-10 (12(a)), we observed that the undefended model had an ASR of 40%. All *BP* and *PadNet* models had an ASR of 0% for all ϵ values. Note that these results hold even when we use only mean and uniform padding, so they do not depend on PGD padding. Other defenses had between 10%–37% ASR for $\epsilon = 0$, but these rose to between 72%–100% for $\epsilon = 0.9$. We observed similar results on the MNIST dataset as shown in Figure 11(a).

Figures 11(b) and 12(b) show ASRs with increasing *confidence* on the x-axis for all models against the Carlini & Wagner l_2 white-box attack on MNIST and CIFAR-10, respectively. When confidence zero on CIFAR-10 (12(b)), we observed an ASR of 92%–96% for the undefended model, MixUp, PGD adversarial training, gradient regularization, and TRADES, compared with 25%–31% for *BP* and 22%–23% for *PadNet* models. *PadNet* had a 12% to 26% improvement compared to *BP*, showing the benefits of *TGR*. When confidence was set to 20, we observed the same ASR for all other models, while the *BP* and *PadNet* models had 0% to 2% ASR. We observed similar results on the MNIST dataset shown in Figure 11(b), and for the Carlini & Wagner l_∞ white-box attacks on CIFAR-10 shown in Figure 12(c).

Figures 11(d) and 12(d) show ASR with increasing ϵ on the x-axis for all models against the Deepfool white-box attack on MNIST and CIFAR-10, respectively. For all ϵ values, *BP* and *PadNet* maintained 0% to 6% ASR on MNIST, and 11% to 19% on CIFAR-10. MixUp showed mixed results with 6% ASR on MNIST and 97% on CIFAR-10. In contrast, all other models had 91%–94% ASR on CIFAR-10, and 40%–100% on MNIST.

Our results indicate that *BP* and *PadNet* provide significant robustness compared to the other defenses. Moreover, for several of the attack settings we tested, the other defenses appeared to be no better than the undefended model. In most cases, *PadNet* significantly improves robustness compared to *BP*, highlighting the importance of *TGR* in conjunction with *BP* augmentation. Table 6 shows similar results for black-box attacks on MNIST and CIFAR-10.

Table 5. Defensive Models and Their Benign Accuracy on CIFAR100 and Tiny ImageNet

Model	CIFAR-100	TinyImageNet Top-1 / Top-5
No-Defense	66.91%	46.56% / 72.75%
BP	70.26%	52.31% / 76.10%
PadNet	69.72%	51.61% / 77.95%
BP-PGD	67.09%	46.55% / 73.70%
PadNet-PGD	65.99%	47.80% / 74.50%

Table 6. Black-Box Adversarial Example ASR on MNIST and CIFAR-10 (**Lower Is Better**)

Defenses	PGD ϵ , 0.1	PGD ϵ , 0.5	PGD ϵ , 0.9	C&W l_2	C&W l_∞	Deepfool
No Defense	0.00	0.05	0.64	0.10	0.03	0.03
MixUp	0.02	0.40	0.75	0.05	0.00	0.01
Madry et. al.[23]	0.00	0.02	0.57	0.00	0.00	0.00
TRADES	0.00	0.49	0.98	0.13	0.00	0.01
Grad. Reg. A	0.02	0.90	0.96	0.07	0.06	0.01
Grad. Reg. B	0.01	0.22	0.96	0.03	0.00	0.02
BP	0.00	0.00	0.00	0.07	0.03	0.06
PadNet	0.00	0.00	0.00	0.07	0.03	0.00
BP w/ PGD	0.00	0.00	0.00	0.07	0.04	0.03
PadNet w/ PGD	0.00	0.00	0.00	0.03	0.03	0.06

Black Box CIFAR-10						
Defenses	PGD ϵ , 0.1	PGD ϵ , 0.5	PGD ϵ , 0.9	C&W l_2	C&W l_∞	Deepfool
No Defense	0.43	0.84	0.84	0.10	0.21	0.09
MixUp	0.27	0.83	0.85	0.10	0.22	0.09
Madry et al.[23]	0.31	0.78	0.85	0.11	0.16	0.11
TRADES	0.14	0.82	0.84	0.18	0.27	0.18
Grad. Reg. A	0.10	0.89	0.90	0.08	0.25	0.08
Grad. Reg. B	0.08	0.74	0.86	0.14	0.19	0.15
BP	0.00	0.00	0.00	0.12	0.06	0.11
PadNet	0.00	0.00	0.00	0.13	0.16	0.14
BP w/ PGD	0.00	0.00	0.00	0.12	0.12	0.12
PadNet w/ PGD	0.00	0.00	0.00	0.11	0.09	0.10

4.5 CIFAR-100 and Tiny ImageNet

We trained and evaluated the PadNet variants on the CIFAR-100 and Tiny ImageNet datasets against PGD, C&W l_2 , C&W l_∞ , and Deepfool. The benign accuracy is shown in Table 5. On CIFAR-100, the undefended model had an accuracy of 66.91%. All PadNet variants, except for PadNet-PGD, increased accuracy by 0.18% to 3.35%, with the best being BP, second being PadNet, and the least improvement being BP-PGD. PadNet-PGD accuracy was reduced by only 0.92%.

For Tiny ImageNet, the undefended model had an accuracy of 46.56% and 72.75% for Top-1 and Top-5, respectively. BP increased accuracy to 52.31% and 76.1% respectively. PadNet increased accuracy to 51.61% and 77.95% for Top-1 and Top-5, respectively. The results from Table 5 indicate that BP and TGR cause regularization and allow the model to generalize better toward data points outside the training set, thus increasing accuracy. We believe there was an increase in accuracy

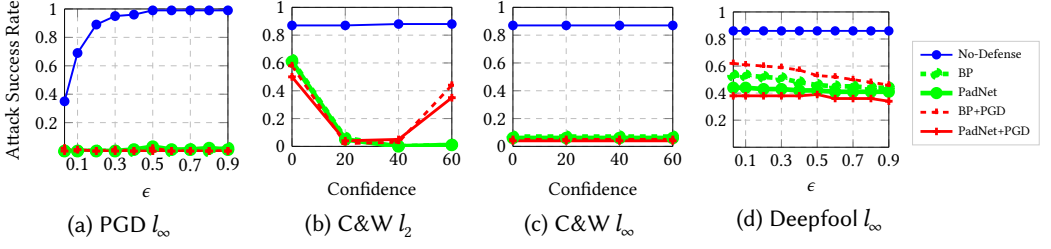


Fig. 13. CIFAR-100: white-box ASR.

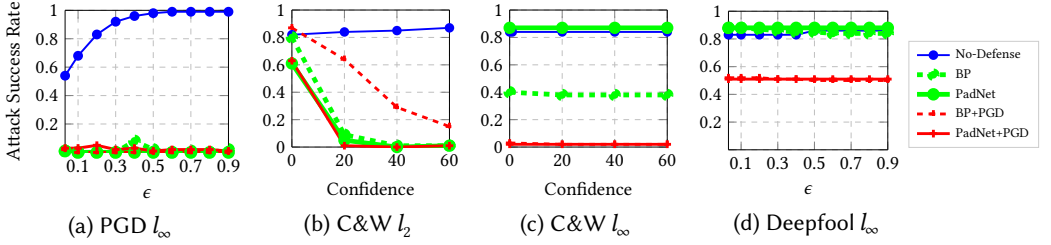


Fig. 14. Tiny-ImageNet: white-box ASR.

for CIFAR-100 and Tiny ImageNet, and not for MNIST and CIFAR-10, due to the increased complexity in the data manifolds, with significantly more classes, which allows more opportunity for regularization and improvement.

4.5.1 CIFAR-100. PadNet robustness on CIFAR-100 is shown in Figure 13. Against the PGD attack (Figure 13(a)), The undefended model had an ASR of 35% for $\epsilon = 0.031$; ASR reached 100% after ϵ was increased beyond 0.4. For all ϵ values, the ASR remained 0% for all PadNet variants.

For C&W l_2 , ASR for PadNet variants was 50%–62% for $conf = 0$, and decreased to 0% for $conf = 20$ and $conf = 40$. For C&W l_∞ , ASR was 0% for all PadNet variants. For Deepfool, the ASR for PadNet variants ranged from 38% to 62%.

4.5.2 Tiny ImageNet. Robustness on Tiny ImageNet is shown in Figure 14. For PGD with $\epsilon = 0.031$, ASR ranged from 1% to 3% for all PadNet variants, and remained near 0% for larger ϵ . For C&W l_2 , the ASR for all PadNet variants except BP-PGD ranged from 61% to 87% for $conf = 0$, decreased to below 10% for $conf = 20$, and remained near 0% for larger confidence values. For C&W l_∞ , ASR was 2% to 3% for BP-PGD and PadNet-PGD, 40% for BP, and 87% for PadNet. For Deepfool, ASR was approximately 51% for BP-PGD and PadNet-PGD, and 84% to 88% for BP and PadNet.

4.6 ImageNet-100 Vision Transformer

ImageNet-100 is a subset of the ImageNet (ILSVRC) dataset containing 100 selected classes out of the full 1,000 ImageNet classes. It is commonly used for experiments that need faster iterations while still maintaining semantic diversity and complexity similar to full ImageNet and is a standard in the literature for efficient benchmarking of model performance, robustness, and adversarial defense research. The dataset contains 100 classes, 390000 training samples, and 5000 test samples of dimensions $224 \times 224 \times 3$.

We trained PadNet and BP on ImageNet-100 using a ViT architecture. The ViT, introduced by Dosovitskiy et al. [10], is a deep learning model that applies the transformer architecture, originally developed for natural language processing, to image classification tasks. ViT segments an image

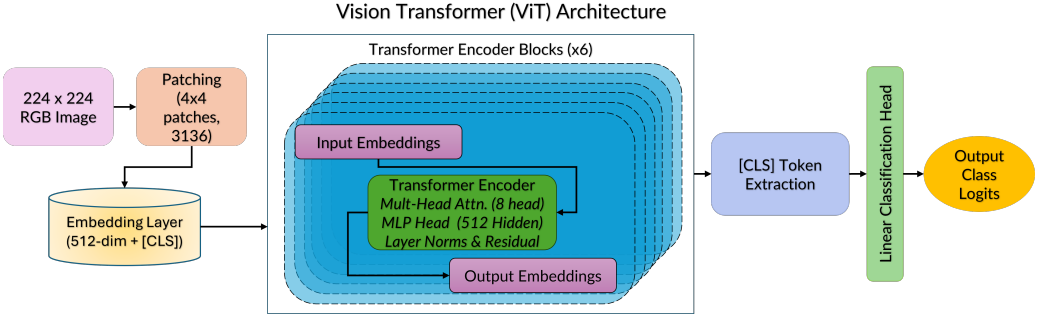


Fig. 15. System architecture of ViT trained with PadNet.

Table 7. Benign Validation Set Accuracy for No-Defense Compared to PadNet and BP Trained on ImageNet100 Using a Vision Transformer Architecture

Model	ImageNet 100
No-Defense ViT	70.86%
No-Defense ViT Transfer	87.72%
BP-ViT	97.42%
PadNet-ViT	99.96%

into a sequence of fixed-size patches, embeds them into a latent space, and processes the sequence with standard transformer blocks, foregoing traditional convolutional operations [10]. Figure 15 showcases our ViT architecture, which divides images into patches of size 16 and includes 6 encoder layers, each with 8 attention heads in the transformer backbone and embedding dimension of 512. The position-wise feedforward network within each encoder uses an MLP with 512 hidden neurons. Both attention and embedding stages were trained with 0.1 dropout to reduce overfitting. The PadNet-ViT was trained with NOTA augmentation using mean and uniform padding samples, and TGR for the loss function. BP-ViT was trained solely with NOTA augmentation.

4.6.1 Benign Accuracy. ViT architectures typically demonstrate that, given sufficient data and compute, transformers can match or exceed the performance of CNNs on vision tasks, especially with transfer learning where weights are pretrained on large-scale datasets like ImageNet-21k [10]. Table 7 shows the benign accuracy results for a vanilla ViT (No-Defense ViT), a ViT trained with transfer learning (No-Defense ViT Transfer), PadNet-ViT, and BP-ViT over 5000 clean test samples. PadNet-ViT significantly outperforms both vanilla and transfer-learned ViTs on ImageNet-100, achieving a striking 99.96% benign accuracy compared to 70.86% for No-Defense ViT and 87.72% for No-Defense Transfer. We believe the improvement is due to PadNet’s strong regularization strategy and its ability to handle hard or atypical samples. While the NOTA augmentation introduces a “none-of-the-above” class, it effectively teaches the model to allocate low confidence to ambiguous or outlier inputs, reducing overfitting and improving generalization. TGR further enhances this effect by penalizing the model’s sensitivity to directions in the input space associated with worst-case classification errors, encouraging smoother, more stable decision boundaries. The BP-ViT variant, which lacks TGR but includes the NOTA class, still performs well (97.42%) by reducing overconfidence on difficult samples and enforcing a more cautious prediction strategy. However, PadNet-ViT outperforms BP-ViT because TGR strengthens the model’s ability to generalize across the entire input space, including edge cases that are often responsible for poor generalization in

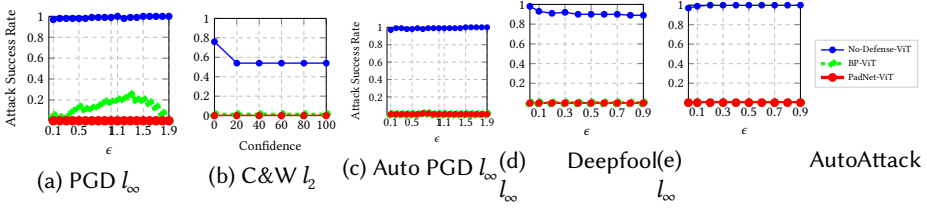


Fig. 16. ImageNet100-ViT: white-box ASR.

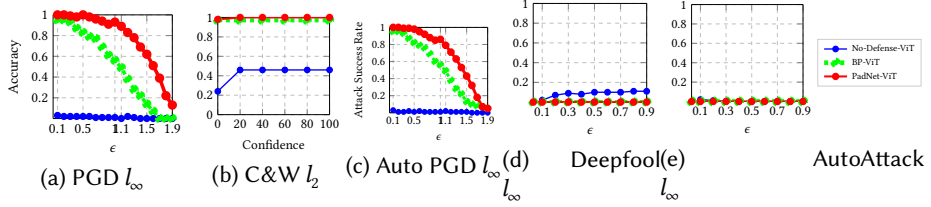


Fig. 17. ImageNet100-ViT: white-box adversarial accuracy.

high-capacity models like ViTs. This combination leads to both improved fit to the training data and better performance on the clean validation set.

4.6.2 ImageNet-100 Robustness. We evaluated robustness by attacking the ViT models using five standard attacks: PGD l_∞ , C&W l_2 , Auto PGD l_∞ , Deepfool l_∞ , and AutoAttack l_∞ . ASR quantifies the effectiveness of an attack in causing a model to misclassify an input into an incorrect class, excluding cases where the model correctly predicts the input as belonging to the NOTA class, (i.e., $\hat{y}(x) \neq y \wedge \hat{y}(x) \neq y_{NOTA}$). Figure 16 shows ASR vs. increasing perturbation (ϵ bound or confidence). We found that PadNet-ViT maintained 0 ASR even while increasing perturbation bound for all attacks. BP-ViT showed the same behavior except against PGD where the ASR increased to 0.25 when ϵ was increased beyond 1.2.

Unlike the other architectures that mainly predicted the NOTA class when encountering adversarial examples, the combination of PadNet with ViT allowed the model to achieve high Adversarial Accuracy in many cases. Adversarial Accuracy, shown in Figure 17, is a model's ability to accurately predict the truth label of an adversarial example. PadNet-ViT and BP-ViT achieved 100% adversarial accuracy against C&W l_2 . Similarly, they both scored 100% adversarial accuracy against PGD and Auto PGD while ϵ was less than 0.5, and dropped to near 0% as ϵ reached 1.9. Conversely, adversarial accuracy was 0% for both Deepfool and AutoAttack over all perturbation intensities.

4.6.3 PadNet-ViT Insights. The combination of ViT architecture with PadNet training leads to lower ASR and higher adversarial accuracy compared to other architectures due to the inherent capacity of ViTs to capture global dependencies and the regularization mechanisms introduced by PadNet. ViTs, with their self-attention mechanisms, are particularly well-suited to model subtle patterns across the entire input, which helps to enhance robustness when combined with PadNet's NOTA augmentation and TGR. NOTA helps the model learn to reject adversarial or ambiguous inputs by assigning them to the dedicated NOTA class, which reduces incorrect confident predictions under attack. TGR adds another layer of defense by penalizing directions in input space where the model is most sensitive, encouraging smoother, more stable responses to input variations. This synergy explains why PadNet-ViT achieved 0% ASR across all tested attacks and retained 100% adversarial accuracy under C&W and low- ϵ PGD attacks. In contrast, other architectures trained with PadNet were more prone to predicting NOTA indiscriminately under attack, which, while

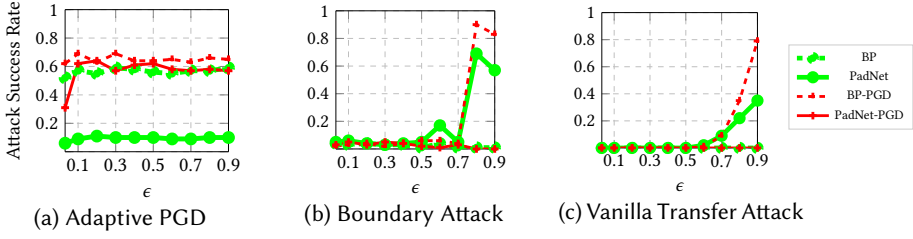


Fig. 18. Adaptive Attacks CIFAR-10: white-box ASR.

lowering adversarial accuracy, still kept ASR low in many cases. The superior performance of PadNet-ViT reflects both the expressive power of ViTs and the effectiveness of PadNet’s principled regularization and rejection mechanisms.

4.7 Adaptive Attacks

In recent years, many proposed defenses appear to exhibit robustness when being evaluating against static or weak attacks. Tramèr et al. [41] show that adaptive attacks, i.e., attacks specifically designed to target a given defense, cause substantially reduced robustness compared to the original claims. Therefore, we attack PadNet with three adaptive attacks. Figure 18 shows ASR on the CIFAR-10 dataset. First, we design an adaptive attack based on the PGD approach, in which a sample is taken from the l_p -ball of radius ϵ centered at x , followed by adjusting x incrementally by adding the gradient $\nabla_x L(F(x), y)$ with step size α bounded by ϵ at every iteration. Two key characteristics of PadNet are: (1) the dataset is augmented with padding class examples labeled as y_{pad} , and (2) TGR regularizes the gradient $\nabla_x L(F(x), y_{pad})$ within the cost function. Therefore, we create a unique adaptive attack by substituting the gradient used in PGD with $\nabla_x L(F(x), y_{pad})$. This causes PGD to move towards the decision boundary as indicated by the padding class, and potentially through it. For the adaptive PGD attack, ASR for PadNet ranged from .06 to .1 for all ϵ values. For the other PadNet variants, ASR ranged from .31 to .37 for $\epsilon = 0.031$, and increased to .41 to .62 as ϵ was increased.

Next, Tramèr et al. [40] suggested that *decision-based attacks* can be used to detect gradient masking, because the attack does not rely on gradient information to find adversarial examples. They show that models trained against one l_p -norm learn to mask gradients for attacks in other l_p -norms, giving a false sense of security. To show that PadNet does not exhibit gradient masking, we attack with a *decision-based* attack called the *Boundary Attack* [4]. As shown in Figure 18(b), ASR for all PadNet variants against the Boundary attack are .02 to .2 for ϵ less than .7.

Finally, Athalye et al. [2] recommended another test for detecting gradient masking: applying a transfer attack in which adversarial examples are generated on a vanilla undefended model and then evaluated on the defended model. They successfully use this strategy to break models that rely on gradient masking, or to estimate gradients for non-differentiable models such as stochastic classifiers. As shown in Figure 18(c), the ASR for a vanilla transfer attack was near 0% for all PadNet variants while ϵ was less than .6 on CIFAR-10.

Based on these findings, we see that: (1) an adaptive attack that attempts to use the padding class as a signal for where to push through the decision boundary is somewhat effective except for vanilla PadNet, and (2) PadNet and TGR do not exhibit the weaknesses of approaches based on gradient masking.

Finally the **Backward Pass Differentiable Approximation (BPDA)** attack was designed to overcome defenses that rely on non-differentiable transformations or masked gradients by substituting these operations with differentiable approximations during backpropagation [2]. However,

PadNet does not rely on gradient obfuscation; its gradients are fully differentiable. Moreover, PadNet was explicitly evaluated against the Boundary Attack and the Vanilla Transfer Attack, which similarly bypass gradient masking and non-differentiability, thereby addressing the same concerns that BPDA is intended to exploit.

5 Discussion

While PadNet demonstrates strong performance in both clean and adversarial settings, it is important to recognize the practical limitations and challenges that remain. Despite achieving low ASRs and high adversarial accuracy in controlled evaluations, real-world deployments may still encounter unexpected or adaptive adversarial scenarios. In such cases, PadNet's use of the NOTA class provides a valuable fallback – flagging uncertain or anomalous inputs rather than forcing potentially incorrect class predictions. However, this also places a burden on system designers and users to define appropriate actions when NOTA predictions occur, such as deferring to human review, triggering additional verification mechanisms, or logging for offline analysis. More broadly, defending against adversarial examples remains a fundamentally hard problem due to the complexity of modern models and the high dimensionality of input spaces, which give rise to vast and fragile decision boundaries. While PadNet makes meaningful progress by encouraging caution and improving generalization near these boundaries, it cannot fully eliminate the inherent vulnerabilities of deep learning systems. As such, a layered defense strategy – combining architectural, training, and procedural safeguards – remains essential for deploying robust AI systems in practice.

5.1 PadNet Limitations

While PadNet, along with its components NOTA augmentation and TGR, offers strong performance and rejection capabilities, it also introduces several limitations. First, incorporating a NOTA class requires careful calibration of decision thresholds and sufficient coverage of outlier or ambiguous inputs during training; otherwise, the model may over-rely on NOTA and underperform on challenging in-distribution examples. This can lead to over-rejection, especially in settings with distribution shift or underrepresented subclasses. Additionally, TGR increases training complexity and computational cost, as it involves computing targeted adversarial gradients and incorporating them into the loss function. This gradient-based regularization may also limit scalability to very large datasets or real-time applications. Moreover, PadNet's effectiveness can be architecture-dependent: while ViTs benefit from its structure, other models may show degraded clean accuracy or unstable NOTA predictions when subjected to adversarial noise. Finally, designing a balanced and representative NOTA training set is nontrivial, as overly aggressive or unrepresentative augmentations can inadvertently shift the model's focus away from fine-grained distinctions between valid classes.

5.2 Computational Overhead

The computational overhead of PadNet arises primarily from the BP and TGR components. In BP, the training dataset is effectively tripled since, for each input, one mean-padding sample and one uniform-padding sample are generated dynamically and included in the training batch. This increases both memory usage and per-epoch training time in proportion to the augmented batch size. With BP-PGD, the cost is higher because an additional adversarial example counterpart generated via PGD is included, effectively quadrupling the dataset and requiring additional forward-backward passes for adversarial sample generation. Beyond data augmentation, TGR introduces further overhead by adding a gradient-based regularization term to the loss function, which requires computing an adversarial direction toward the padding class and the associated ℓ_2 -norm penalty. While these additions increase training time relative to standard baselines, the costs are fixed; every sample contributes a bounded number of augmented counterparts, and they occur only during

training. Importantly, PadNet introduces no additional overhead during inference or deployment, as all augmentations and gradient penalties are used exclusively in the training stage.

5.3 When PadNet is Attacked

Given the approach that *BP* and *PadNet* represent, i.e., introducing a padding or “none-of-the-above” class, it is further significant to consider how or whether the presence of such controls may alter the behavior of the defended systems. In other defenses, attacks go unnoticed as silent errors because the models attempt to classify any input as a known class. In PadNet, adversarial examples are classified as the padding class when the defense is successful. Therefore, it is important to consider what is to be done with an input when it is found to be in the padding class. This may be very useful in human-machine teaming as the operator will have some indication that the system is being attacked, and thus may take the necessary response actions. In other challenging environments, the models may experience a shift in distribution. For example, some model might be trained in an environment that consists of good weather. If there is a shift in weather, an undefended model may experience errors while a PadNet model should produce padding class predictions which may serve as warnings to an operator. Again context is key in determining the right default course of action. What is “safe” or “conservative” in one context may not be sufficient or even appropriate in another.

5.4 Generalizing Across Data Sparse Regions

Complex datasets have a vast sample space in which the data points within the dataset only occupy a tiny fraction. In other words, the ratio of feature space to sample space is very small. For example, the MNIST dataset contains 60,000 28x28 pixel images existing in a feature space of 784 dimensions. Thus, the sample space Ω is 256^{784} . Moreover, the data sparse regions in between and around class distributions is also very large. It is not our goal to fully occupy all data sparse regions with barrier class samples as this would be NP complete. Instead, we augment the training set by a factor of 3 and rely on the neural network to generalize across the remaining data sparse regions to produce padding class predictions.

6 Conclusion

We have presented PadNet, a defense against adversarial examples based on building a barrier using a combination of training samples that mix multiple classes together. PadNet uses the barrier to separate decision boundaries between benign classes with regions of padding, and then implements a gradient regularization strategy that penalizes gradients in the direction of the padding, causing the decision boundary to draw tighter around training samples and introducing a thicker boundaries between classes. We evaluate PadNet against a sampling of the most effective state-of-the-art attacks, demonstrating that it offers strong robustness compared to the defenses that were tested in the evaluation.

Acknowledgments

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the U.S. Naval Academy, the Department of the Navy, the Department of War, or the U.S. Government.

References

- [1] Ahmed Aldahdooh, Wassim Hamidouche, Sid Ahmed Fezza, and Olivier Déforges. 2022. Adversarial example detection for DNN models: A review and experimental comparison. *Artificial Intelligence Review* 55, 6 (Aug. 2022), 4403–4462. DOI: <https://doi.org/10.1007/s10462-021-10125-w>
- [2] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. arXiv:1802.00420. Retrieved from <https://arxiv.org/abs/1802.00420>

- [3] Reza Bayat and Irina Rish. 2024. Adversarial training with synthesized data: A path to robust and generalizable neural networks. *OpenReview (ICML 2024 Workshop – Next Gen AI Safety)* (2024). Retrieved from <https://openreview.net/pdf?id=H6V1NW7bGS>
- [4] Wieland Brendel, Jonas Rauber, and Matthias Bethge. 2017. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. arXiv:1712.04248. Retrieved from <https://arxiv.org/abs/1712.04248>
- [5] Xiaoyu Cao and Neil Zhenqiang Gong. 2017. Mitigating evasion attacks to deep neural networks via region-based classification. arXiv:1709.05583. Retrieved from <https://arxiv.org/abs/1709.05583>
- [6] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 39–57.
- [7] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2018. Adversarial attacks and defences: A survey. arXiv:1810.00069 [cs.LG]. Retrieved from <https://arxiv.org/abs/1810.00069>
- [8] Terrance DeVries and Graham W. Taylor. 2017. Improved regularization of convolutional neural networks with cutout. arXiv:1708.04552. Retrieved from <https://arxiv.org/abs/1708.04552>
- [9] Junhao Dong, Seyed-Mohsen Moosavi-Dezfooli, Jianhuang Lai, and Xiaohua Xie. 2022. The enemy of my enemy is my friend: Exploring inverse adversaries for improving adversarial training. arXiv:2211.00525. Retrieved from <https://arxiv.org/abs/2211.00525>
- [10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv:2010.11929. Retrieved from <https://arxiv.org/abs/2010.11929>
- [11] H. Drucker and Y. Le Cun. 1992. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks* 3, 6 (1992), 991–997. DOI : <https://doi.org/10.1109/72.165600>
- [12] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2020. Sharpness-aware minimization for efficiently improving generalization. arXiv:2010.01412. Retrieved from <https://arxiv.org/abs/2010.01412>
- [13] Salah Ghamizi, Jingfeng Zhang, Maxime Cordy, Mike Papadakis, Masashi Sugiyama, and Yves Le Traon. 2023. GAT: Guided adversarial training with Pareto-optimal auxiliary tasks. arXiv:2302.02907. Retrieved from <https://arxiv.org/abs/2302.02907>
- [14] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations (ICLR'15)*.
- [15] Sven Gowal, Chongli Qin, Po-Sen Huang, Taylan Cemgil, Krishnamurthy Dvijotham, Timothy Mann, and Pushmeet Kohli. 2020. Achieving robustness in the wild via adversarial mixing with disentangled representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1211–1220.
- [16] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. 2020. Uncovering the limits of adversarial training against norm-bounded adversarial examples. arXiv:2010.03593. Retrieved from <https://arxiv.org/abs/2010.03593>
- [17] Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019. Mixup as locally linear out-of-manifold regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3714–3722.
- [18] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. 2018. Adversarial logit pairing. arXiv:1803.06373. Retrieved from <https://arxiv.org/abs/1803.06373>
- [19] A. Kherchouche, S. A. Fezza, Wassim Hamidouche, and O. Déforges. 2020. Detection of adversarial examples in deep neural networks with natural scene statistics. In *Proceedings of the 2020 International Joint Conference on Neural Networks, IJCNN 2020 (Proceedings of the International Joint Conference on Neural Networks)*. Institute of Electrical and Electronics Engineers Inc., Glasgow, United Kingdom, 9206959. DOI : <https://doi.org/10.1109/IJCNN48605.2020.9206959>
- [20] Matan Levi and Aryeh Kontorovich. 2024. Splitting the difference on adversarial training. In *Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24)*. USENIX Association, Philadelphia, PA, 3639–3656. Retrieved from <https://www.usenix.org/conference/usenixsecurity24/presentation/levi>
- [21] Shiqing Ma, Yingqi Liu, Guanhong Tao, Wen-Chuan Lee, and Xiangyu Zhang. 2019. NIC: Detecting adversarial samples with neural network invariant checking. In *Proceedings of the 2019 Network and Distributed System Security Symposium*. Internet Society, San Diego, CA. DOI : <https://doi.org/10.14722/ndss.2019.23415>
- [22] Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E. Houle, and James Bailey. 2018. Characterizing adversarial subspaces using local intrinsic dimensionality. arXiv:1801.02613. Retrieved from <https://arxiv.org/abs/1801.02613>
- [23] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. arXiv:1706.06083. Retrieved from <https://arxiv.org/abs/1706.06083>
- [24] Thibault Maho, Teddy Furon, and Erwan Le Merrer. 2021. SurFree: A fast surrogate-free black-box attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. arXiv:2011.12807. Retrieved from <https://arxiv.org/abs/2011.12807>

- [25] Dongyu Meng and Hao Chen. 2017. MagNet: A two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS'17)*. ACM, New York, NY, USA, 135–147. DOI : <https://doi.org/10.1145/3133956.3134057>
- [26] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: A simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2574–2582.
- [27] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, et al. 2018. Adversarial robustness toolbox v1. 0.0. arXiv:1807.01069. Retrieved from <https://arxiv.org/abs/1807.01069>
- [28] Tianyu Pang, Kun Xu, and Jun Zhu. 2019. Mixup inference: Better exploiting mixup to defend adversarial attacks. arXiv:1909.11515. Retrieved from <https://arxiv.org/abs/1909.11515>
- [29] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. 2016. Towards the science of security and privacy in machine learning. arXiv:1611.03814. Retrieved from <https://arxiv.org/abs/1611.03814>
- [30] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 582–597.
- [31] Jeonghwan Park, Paul Miller, and Niall McLaughlin. 2024. Hard-label based small query black-box adversarial attack. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 3986–3995.
- [32] Leo Hyun Park, Jaekuk Kim, Myung Gyo Oh, Jaewoo Park, and Taekyoung Kwon. 2024. Adversarial feature alignment: Balancing robustness and accuracy in deep learning via adversarial training. In *Proceedings of the 2024 ACM Workshop on Artificial Intelligence and Security (AISec'24), co-located with ACM CCS 2024*. arXiv:2402.12187. Retrieved from <https://arxiv.org/abs/2402.12187>
- [33] Jie Qin, Jiemin Fang, Qian Zhang, Wenyu Liu, Xingang Wang, and Xinggang Wang. 2020. Resizemix: Mixing data with preserved object information and true labels. arXiv:2012.11101. Retrieved from <https://arxiv.org/abs/2012.11101>
- [34] Han Qiu, Yi Zeng, Tianwei Zhang, Yong Jiang, and Meikang Qiu. 2020. Fencebox: A platform for defeating adversarial examples with data augmentation techniques. arXiv:2012.01701. Retrieved from <https://arxiv.org/abs/2012.01701>
- [35] Andrew Slavin Ross and Finale Doshi-Velez. 2018. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.
- [36] Ali Shafahi, W. Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. 2019. Are adversarial examples inevitable?. In *Proceedings of the International Conference on Learning Representations*. Retrieved from <https://openreview.net/forum?id=r1IWUoA9FQ>
- [37] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. arXiv:1312.6199. Retrieved from <https://arxiv.org/abs/1312.6199>
- [38] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR'13)*.
- [39] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. 2020. Measuring robustness to natural distribution shifts in image classification. arXiv:2007.00644. Retrieved from <https://arxiv.org/abs/2007.00644>
- [40] Florian Tramèr and Dan Boneh. 2019. Adversarial training and robustness for multiple perturbations. *Advances in Neural Information Processing Systems* 32 (2019).
- [41] Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. 2020. On adaptive attacks to adversarial example defenses. arXiv:2002.08347. Retrieved from <https://arxiv.org/abs/2002.08347>
- [42] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2018. Ensemble adversarial training: Attacks and defenses. In *Proceedings of the International Conference on Learning Representations (ICLR'18)*.
- [43] A. F.M. Shahab Uddin, Mst. Sirazam Monira, Wheemyung Shin, TaeChoong Chung, and Sung-Ho Bae. 2021. SaliencyMix: A saliency guided data augmentation strategy for better regularization. In *Proceedings of the International Conference on Learning Representations*. Retrieved from <https://openreview.net/forum?id=-M0QkvBGTTq>
- [44] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, 11 (2008).
- [45] Viet Quoc Vo, Ehsan Abbasnejad, and Damith C. Ranasinghe. 2024. BruSLAttack: A query-efficient score-based black-box sparse adversarial attack. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*. Retrieved from <https://arxiv.org/abs/2404.05311>
- [46] Zekai Wang, Tianyu Pang, Chao Du, Min Lin, Weiwei Liu, and Shuicheng Yan. 2023. Better diffusion models further improve adversarial training. In *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*. arXiv:2302.04638. Retrieved from <https://arxiv.org/abs/2302.04638>

- [47] Yaoqing Yang, Rajiv Khanna, Yaodong Yu, Amir Gholami, Kurt Keutzer, Joseph E. Gonzalez, Kannan Ramchandran, and Michael W. Mahoney. 2020. Boundary thickness and robustness in learning models. *Advances in Neural Information Processing Systems* 33 (2020), 6223–6234.
- [48] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6023–6032.
- [49] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide residual networks. arXiv:1605.07146. Retrieved from <https://arxiv.org/abs/1605.07146>
- [50] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization. arXiv:1710.09412. Retrieved from <https://arxiv.org/abs/1710.09412>
- [51] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. 2019. Theoretically principled trade-off between robustness and accuracy. In *Proceedings of the International Conference on Machine Learning*. PMLR, 7472–7482.

Received 12 February 2025; revised 24 September 2025; accepted 5 November 2025