2019 IEEE European Symposium on Security and Privacy (EuroS&P)

Towards Understanding Limitations of Pixel Discretization Against Adversarial Attacks

Jiefeng Chen [1] Xi Wu [2] Vaibhav Rastogi [1] Yingyu Liang [1] Somesh Jha [1]
[1] University of Wisconsin-Madison [2] Google

***Abstract*—Wide adoption of artificial neural networks in various domains has led to an increasing interest in defending adversarial attacks against them. Preprocessing defense methods such as pixel discretization are particularly attractive in practice due to their simplicity, low computational overhead, and applicability to various systems. It is observed that such methods work well on simple datasets like MNIST, but break on more complicated ones like ImageNet under recently proposed strong white-box attacks. To understand the conditions for success and potentials for improvement, we study the pixel discretization defense method, including more sophisticated variants that take into account the properties of the dataset being discretized. Our results again show poor resistance against the strong attacks. We analyze our results in a theoretical framework and offer strong evidence that pixel discretization is unlikely to work on all but the simplest of the datasets. Furthermore, our arguments present insights why some other preprocessing defenses may be insecure.**

I. Introduction

Deep learning models have been shown to be vulnerable to adversarial perturbations, which are slight perturbations of natural input data but are classified differently than the natural input [1]. Adversarial robustness, or reducing this vulnerability to adversarial perturbations, of deep neural networks (DNNs) has received significant attention in recent years ([2], [3], [4], [5], [6], [7], [8], [9]), due to the interest of deploying deep learning systems in various security-sensitive or security-critical applications, such as self-driving cars and identification systems. Among various methods, defense methods that preprocess the input and then pass them to the existing learning systems are particularly attractive in practice due to their simplicity, low computational overhead, and direct applicability to various neural network systems without interfering with the training. Various such preprocessing methods have been proposed and achieve good results against some old attacks. Unfortunately, as observed by Athalye et al. [10], when applied to typical datasets, these methods fall in front of the stronger Backward Pass Differentiable Approximation attack (BPDA). It is now a general perception that preprocessing methods cannot work against such strong attacks. However, the question about why they fail remain largely open.

This paper studies a prototypical preprocessing defense method called *pixel discretization*, taking a first step towards understanding the effectiveness and limitations of such methods. Pixel discretization works by first constructing a codebook of codewords (also called codes) for pixel values, then discretizing an input vector using these codewords, and finally feeding the discretized input to a pre-trained base model, with the hope that adversarial noise can be removed by the discretization. For example, the color-depth reduction by Xu et al. [5] and the subsequent thermometer encoding technique proposed by [6] essentially "round" pixels to nearby codewords during preprocessing. They achieve good results on simple datasets such as MNIST [11] against strong attacks such as Carlini-Wagner [12], but later are shown to achieve poor performance against BPDA when applied to datasets of moderate or higher complexity (such as CIFAR-10 [13]). However, we observe that both of these pixel discretization proposals employ very simple codes which are *completely independent of the data distribution*. This thus leads to the following natural and intriguing question:

> *Are there more sophisticated* pixel discretization *techniques, especially those that are data-specific, which would be significantly better against white-box attacks?*

This paper studies the effectiveness of pixel discretization techniques, including data-specific ones, through extensive theoretical and empirical analyses. Our main results suggest a strong negative answer for the above question: if the underlying base model has poor adversarial robustness, pixel discretization is unlikely to improve robustness except for very simple datasets. On one hand, we obtain strong performance on simple datasets like MNIST, identify conditions under which the method can provably work, and propose a simple certificate that can be computed to lower bound the performance against any attacks within a budget. On the other hand, poor performance is observed on more complicated datasets like CIFAR-10. We then provide detailed analysis on why the method fails on such data.

A primary tenet of our study is to view discretization by codewords as constructing a clustering of pixels, where pixels in the same cluster gets assigned to the same codeword. Our key observation is a stringent tradeoff between accuracy and robustness: To allow only a marginal reduction in accuracy, we would like to keep our clusters of pixels small so that only close enough pixels map to cluster centers. However, on complex data this requires many small, closely-packed clusters whose boundary pixels can be easily perturbed to pixels in other clusters, implying that robustness will not be high enough if the underlying base model has poor adversarial robustness.

As a result, our theoretical and empirical analyses hinge on the separability of pixels: If the pixels form well-separated clusters, we can easily map them to cluster centroids and achieve a discretization where pixel perturbation does not change the clusters they belong to, implying adversarial robustness. Based

© 2019, Jiefeng Chen. Under license to IEEE.
DOI 10.1109/EuroSP.2019.00042

IEEE computer society

on this understanding, we provide a framework for formally certifying adversarial robustness. While our theory is able to provide certificates of robustness for simple datasets such as MNIST, for more complex datasets like CIFAR-10, we show by various means that pixel discretization is an inadequate defense against adversarial examples. More specifically, complex datasets do not form well-separated clusters, so pixel discretization is unlikely to improve adversarial robustness.

Many of our arguments are general and may be used to study other preprocessing techniques as well. For instance, if a defense is not performed at the pixel level but at the level of patches of pixels, our arguments will extend to that defense as well.

In summary, this paper makes the following contributions:

- We develop more sophisticated pixel discretization techniques that take into account the structure of the data points for discretization. Our experiments show that even these sophisticated defenses do not perform well against BPDA attacks.
- We develop a theoretical framework for pixel discretization defenses and present conditions under which they are robust.
- We confirm the general expectation that pixel discretization works well on simple datasets like MNIST. In fact, we are able to formally certify that pixel discretization on such datasets is robust against any adversarial attack. This also shows that gradient masking preprocessing defenses are possible for some simple distributions.
- We further analyze complex datasets like CIFAR-10 to argue that it is unlikely to achieve robustness with pixel discretization on these datasets.

The rest of this paper is organized as follows. Section II presents the necessary background. We then present our experiments with color-depth reduction defense in Section III. We then explore data-specific pixel discretization defenses in Section IV. In Section V, we present our theoretical framework for analyzing the results as well as present further empirical results on MNIST and CIFAR-10 supporting the analysis. We present a discussion with insights on preprocessing defenses in general in Section VI. Finally, we present related work in Section VII and conclude in Section VIII.

The code for the experiments is available at https://github.com/jfc43/pixel-discretization.

II. Preliminaries

This section presents the relevant background for this work, important definitions, and the datasets used in our evaluations.

A. Background

Deep neural network (DNN) models are often vulnerable to adversarial inputs, which are slight perturbations of "natural" test inputs but lead the model to produce adversary-selected outputs different from the outputs to the natural inputs. The susceptibility of neural networks to adversarial perturbations was first discovered by [14], [1], and a large body of research has been devoted to it since then. A long line of recent research is dedicated to attacks with and defenses against adversarial perturbations. Several other types of attacks have been explored in the context of machine learning. These include *training-time* attacks where an adversary wishes to poison a data set so that a "bad" hypothesis is learned by an ML-algorithm; *model extraction* attacks where an attacker learns a model by making queries to an available model; and a *model inversion* attack where the attacker recovers the inputs used to train a model. The reader is referred to Section VII for a very brief survey of these research works.

In this paper, we focus on *test-time* attacks on supervised image classification problems. The input image $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$(the image has H rows, W columns and C channels). $\mathbf{x}[i,j,l]$ represents the value of i_{th} row, j_{th} column and l_{th} channel of the image. $\mathbf{x}[i,j]$ represents the pixel at i_{th} row and j_{th} column. We have a DNN model F_θ (where θ are model parameters, and we simply write F if θ is clear from the context) that maps $\mathbb{R}^{H\times W\times C}$ to a set of class labels. In our context we assume that the classifier F has been trained without any interference from the attacker (i.e. no training time attacks). Given an attack distance metric $\|\cdot\|$, the goal of an attacker is to craft a perturbation δ so that $F(\mathbf{x}+\delta) \neq F(\mathbf{x})$, where $\mathbf{x} \sim \mathcal{D}$ and $\|\delta\| \leq \varepsilon$. There are several algorithms for crafting the perturbation δ, but in this paper we use the recent method by [10].

There have been several directions for defenses to test-time attacks, but we mainly focus on preprocessing defenses. In such defenses, one designs a preprocessing function $T : \mathbb{R}^{H\times W\times C} \mapsto \mathbb{R}^{H\times W\times C}$, and with a base model F, the end-to-end predictions are produced as $F(T(\cdot))$. In this context there are three types of attacks: (1) *Black Box Attack* where the attacker can only get zero-order information of F (i.e. the outputs of F correspond to the given inputs), (2) *Gray Box Attack* where the attacker knows F, but is unaware of T, and (3) *White Box Attack* where the attacker knows both F and T. This paper considers white-box attacks since this is the strongest attack model.

Given a preprocessing technique T, the goal of an adversary is thus to find another image $\mathbf{z}$ such that $\|\mathbf{z}-\mathbf{x}\| \leq \varepsilon$ and $F(T(\mathbf{z})) \neq F(T(\mathbf{x}))$.

B. Backward Pass Differentiable Approximation (BPDA) Attack

Introduced recently by Athalye et al. [10], the *backward pass differentiable approximation (BPDA)* attack is a white-box attack that assumes the knowledge of both the classifier F and the preprocessor T. The BPDA-attack, given an input and a target label, computes adversarial perturbations by gradient descent over $F(T(\cdot))$. The main problem that BPDA solves is that many preprocessing techniques "obfuscate gradients" by using a T that is not differentiable or is random. BPDA addresses this issue by using $F(T(\cdot))$ for the forward pass but switching to a "differentiable approximation" of T when computing gradients for the backward pass.

C. Adversarial Training

We briefly describe the technique of adversarial training. For detailed account the reader is referred to [15]. Roughly speaking adversarial training works as follows: before processing data $\mathbf{x}$ (e.g. using it in an iteration of stochastic-gradient descent (SGD)) we apply an algorithm for crafting adversarial examples, such as BPDA, and obtain an adversarial example $\mathbf{x}^*$. The learning algorithm uses $\mathbf{x}^*$ instead of $\mathbf{x}$. The theoretical underpinnings of adversarial training can be found in robust optimization.

D. Metrics and Definitions

We now provide the necessary definitions that are used in our work.

Definition 1 (Accuracy)**.** *Accuracy of a classifier F is defined as*

$$\Pr_{(\mathbf{x},y)\sim\mathcal{D}}[F(\mathbf{x}) = y]$$

Definition 2 (Local robustness predicate)**.** *Given an image $\mathbf{x}$, the condition for an adversary with budget ε to not succeed is defined as the following predicate:*

$$\mathcal{R}_\varepsilon(\mathbf{x}, y) \equiv \{F(T(\mathbf{z})) = y \text{ for any } \|\mathbf{z} - \mathbf{x}\| \leq \varepsilon\}$$

We call this predicate the robustness predicate.

Definition 3 (Local certificate)**.** *A predicate $P(\mathbf{x}, y)$ is called a local certificate iff $P(\mathbf{x}, y)$ implies $\mathcal{R}_\varepsilon(\mathbf{x}, y)$. In other words, if $P(\mathbf{x}, y)$ is true then it provides a proof of the robustness at $\mathbf{x}$.*

Definition 4 (Robustness accuracy)**.** *The following quantity is called robustness accuracy*

$$\Pr_{(\mathbf{x},y)\sim\mathcal{D}}[F(T(\mathbf{z})) = y \text{ for any } \|\mathbf{z} - \mathbf{x}\| \leq \varepsilon] = \Pr_{(\mathbf{x},y)\sim\mathcal{D}}[\mathcal{R}_\varepsilon(\mathbf{x}, y)]$$

Robustness accuracy is the measure of robustness across the entire probability distribution. This quantity is the accuracy under the strongest attack. However, we cannot measure it directly in experiments. We instead measure the accuracy under the attacks we perform as an estimation of robustness accuracy.

In general, any norm can be used for the attack distance metric$\|\cdot\|$ used in the definitions above. We specifically use the ℓ_∞ norm ($\|\cdot\|_\infty$) for all discussions and results in this paper. The ℓ_∞ norm is defined as

$$\|\mathbf{u}\|_\infty = \max_i |u_i|$$

where $\mathbf{u}$ is a vector $(u_1, \cdots, u_n)$. Throughout this paper, any mention of the norm means ℓ_∞ norm.

E. Datasets

We use a variety of simple and complex datasets to understand the interaction of pixel discretization and adversarial attacks.

a) MNIST and Fashion-MNIST: The MNIST dataset [11] is a large dataset of handwritten digits. Each digit has 5,000 training images and 1,000 test images. Each image is a 28x28 grayscale with the pixel color-depth of 8 bits.

The MNIST dataset is largely considered a simple dataset, so we also consider a slightly more complex dataset called Fashion-MNIST [16]. Images in this dataset depicts wearables such as shirts and boots instead of digits. The image format, the number of classes, as well as the number of examples are all identical to MNIST.

b) CIFAR-10: The CIFAR-10 dataset [13] is also a dataset of 32x32 color images with ten classes, each consisting of 5,000 training images and 1,000 test images. The classes correspond to dogs, frogs, ships, trucks, etc.

c) GTSRB: The German Traffic Sign Recognition Benchmark (GTSRB) [17] is a dataset of color images depicting 43 different traffic signs. The images are not of a fixed dimensions and have rich background and varying light conditions as would be expected of photographed images of traffic signs. There are about 39,000 training images and 12,000 test images.

Some of the images in the dataset are very dark, making it difficult to identify the right features in the images. This makes adversarial perturbations on them easier. Therefore, we remove all images that have an average intensity of less than 50.

d) ImageNet: ImageNet [18] is a complex dataset with over 14 million images and twenty thousand categories. We use the subset of ImageNet used by the NIPS Adversarial Attacks & Defenses Challenge [19], which contains 1000 development images and 5000 test images from 1001 categories.

F. Experiment Settings

a) Pre-trained Models: For MNIST and CIFAR-10, we use naturally and adversarially pre-trained models from [15]. For ImageNet, we use a naturally pre-trained InceptionResNet-V2 model from [20] and an adversarially pre-trained InceptionResNet-V2 model from [21].

b) Training Hyper-parameters.: For MNIST and CIFAR-10, we use the same training hyper-parameters as [15]. To train models on Fashion-MNIST, we use the same hyper-parameters as those we use to train MNIST. And the same hyper-parameters as those used on CIFAR-10 are used to train models on GTSRB. In order to reduce training time, when we naturally (or adversarially) retrain models, we use naturally (or adversarially) pre-trained model to initialize model parameters and train for 10000 epochs.

c) Attack Methods: If gradient approximation is needed, we use the BPDA attack, otherwise we use the PGD attack. For MNIST and Fashion-MNIST, we set $\varepsilon = 0.3$ and use 100 steps for the attack. For CIFAR-10, when we use naturally trained model as base model, we set $\varepsilon = 2$ and use 40 steps attack; when we use adversarially trained model, we set $\varepsilon = 8$ and use 40 steps. For GTSRB, we set $\varepsilon = 8$ and use 40 steps. For ImageNet, when we use naturally trained model as based model, we set $\varepsilon = 1$ and use 1 steps attack; when we use

adversarially trained model, we set $\varepsilon = 4$ and use 10 steps for the attack.

III. Robustness of Color-Depth Reduction

We begin by discussing a simple pixel discretization technique as described by Xu et al. [5], a prototypical example of pixel discretization defenses. They describe their techniques as "feature squeezing", which are preprocessing techniques intended to condense a number of features into a smaller number of features. The intuition is that by giving an adversary a smaller space from which to select features, it will make performing an adversarial attack difficult. One of their feature squeezing techniques is color-depth reduction. This is a "simple binning" approach where, for example, a 256-bin color channel (as represented by 8 bits) may be quantized to a 8-bin color channel. Color values are essentially "rounded" to their nearest color bins.

On an image $\mathbf{x}$, we can do k-bin discretization via the following function:

$$T(\mathbf{x})[i,j,l] = \frac{\lfloor (k-1)\,\mathbf{x}[i,j,l] + 0.5 \rfloor}{k-1}$$

Where, $i \in [H]$, $j \in [W]$ and $l \in [C]$. Here, $[H] = \{1, 2, \cdots, H\}$. This assumes that the pixel values of the input image have been scaled to be in $[0, 1]$.

Evaluation. As mentioned by Athalye et al. in their BPDA attack paper, color-depth reduction can be attacked by approximating the preprocessing function T with the identity function. Below we reproduce these results with a more sophisticated differentiable approximation of T and provide details of the robustness on different datasets. We will also use this approximation later in the paper when we discuss more sophisticated data-specific pixel discretization.

In our evaluation of color-depth reduction using the BPDA method, we compute $F(T(\mathbf{x}))$ in the forward pass, while for the backward pass, we replace $T(\mathbf{x})$ by $g(\mathbf{x})$, which is:

$$g(\mathbf{x})[i,j,l] = \frac{\sum_{t=1}^{k} c_t \cdot e^{-\alpha \| \mathbf{x}[i,j,l] - c_t \|}}{\sum_{t=1}^{k} e^{-\alpha \| \mathbf{x}[i,j,l] - c_t \|}}.$$

where $i \in [H]$, $j \in [W]$, $l \in [C]$ and $c_t = \frac{t-1}{k-1}$. Note that when $\alpha \to \infty$, $g(\mathbf{x}) = T(\mathbf{x})$. For MNIST and Fashion-MNIST, we set $\alpha = 10$. For other datasets, we set $\alpha = 0.1$. To evaluate classifier's robustness without discretization, we use the PGD attack. We define natural accuracy, or simply accuracy, as the accuracy on clean data. Similarly, robustness accuracy, which we denote as robustness, is the accuracy under attack.

Results. We present results on MNIST, Fashion-MNIST, CIFAR-10, GTSRB and ImageNet datasets as shown in Table I. Our intention is to see if we can achieve sufficient robustness while maintaining accuracy on these datasets through color-depth reduction.

For MNIST, as Xu et al. showed, we obtain good accuracy with just 2 bins. As can be seen from the table, we obtain a substantial improvement in both the naturally trained model and the adversarially trained model. The results are also very good for Fashion-MNIST. This is somewhat expected as MNIST and Fashion-MNIST are similar datasets.

We also obtain results on CIFAR-10, GTSRB and ImageNet. We present detailed results with different bins. As can be seen from the table, color-depth reduction does not substantially improve robustness under whitebox attacks.

The difference in robustness results for MNIST-like simple datasets and for other datasets is due to the complexity of the datasets. In Section V, we will further investigate this issue. We next try data-specific pixel discretization, which may possibly improve robustness over complex datasets.

IV. Data-specific Pixel Discretization

The fact that the simple binning does not work naturally leads to the question whether one can design more sophisticated discretization schemes with better performance. In particular, a reason why simple binning fails is that it does not take into account the properties of the data. Consider the following intentionally simplified setting: each image has just one pixel x taking values in $[0, 1]$, and images of class 0 lie close to 0.6 and those of class 1 lie close to 1. Using simple binning with codewords $\{0, 1\}$ then fails, as the images will all be discretized to 1. This example then motivates more sophisticated approaches that takes into account the distribution of the data, so as to improve robustness over simple binning by color-depth reduction. Our approach in this section is data-specific: we aim to discretize pixels in a way that takes the density of pixels in the dataset into account. We wish to derive a codebook of pixels from the dataset, which is used by T that replaced each pixel with its nearest codeword under a suitable distance metric.

We begin by describing our framework for codebook construction and then present our experimental results on data-specific discretization.

A. Preprocessing Framework and Codebook Construction

At the high level, our framework is very simple and has the following two steps:

1) At training time, we construct a codebook $\mathcal{C} = \{c_1, c_2, \cdots, c_k\}$ for some k, where c_i is in the pixel space. This codebook $\mathcal{C}$ is then fixed during test time.
2) Given an image $\mathbf{x}$ at test time, T replaces each pixel $\mathbf{x}[i,j]$ with a codeword $c \in \mathcal{C}$ that has minimal distance to $\mathbf{x}[i,j]$.

Intuitively, on one hand, we would like to have a set of codes that are far apart from each other, similar to the requirements of error correcting codes in coding theory, such that even after adversarial perturbation the discretized result will not change. On the other hand, we would like to lose little information so that we can keep a high accuracy after discretization. More precisely, a set of good codes should satisfy the following properties:

- *Separability*. Pairwise distances of codewords is large enough for certain distance metric.

Dataset	Base Model	k	Pre-trained		Re-trained	
			Accuracy	Robustness	Accuracy	Robustness
MNIST	Naturally Trained Model	2	98.76%	75.39%	99.09%	80.91%
		256	99.17%	0.00%	N/A	
	Adversarially Trained Model	2	98.18%	97.32%	98.49%	92.89%
		256	98.40%	92.72%	N/A	
Fashion-MNIST	Naturally Trained Model	2	98.97%	78.05%	99.15%	82.11%
		256	99.10%	0.00%	N/A	
	Adversarially Trained Model	2	98.38%	95.78%	98.74%	92.32%
		256	98.53%	87.38%	N/A	
CIFAR-10	Naturally Trained Model	2	38.30%	18.05%	80.38%	46.44%
		8	83.66%	9.12%	92.27%	28.13%
		16	92.71%	6.72%	94.22%	13.84%
		32	94.40%	4.51%	94.81%	6.10%
		256	95.01%	4.20%	N/A	
	Adversarially Trained Model	2	74.59%	34.30%	78.35%	38.32%
		8	86.51%	46.80%	87.24%	46.89%
		16	87.09%	47.11%	87.65%	47.18%
		32	87.20%	47.20%	87.60%	47.13%
		256	87.25%	45.50%	N/A	
GTSRB	Naturally Trained Model	2	61.91%	32.12%	68.70%	29.60%
		8	95.06%	18.28%	95.39%	18.47%
		16	97.15%	13.88%	97.15%	14.98%
		32	97.34%	9.93%	97.43%	10.02%
		256	97.35%	7.83%	N/A	
	Adversarially Trained Model	2	68.43%	55.55%	70.93%	55.30%
		8	93.08%	74.46%	93.80%	74.90%
		16	93.85%	75.66%	94.96%	76.65%
		32	94.02%	75.45%	94.92%	76.20%
		256	94.11%	74.34%	N/A	
ImageNet	Naturally Trained Model	2	50.80%	24.40%	N/A	
		8	88.10%	30.80%		
		16	92.80%	30.60%		
		32	94.40%	27.30%		
		256	94.50%	27.50%		
	Adversarially Trained Model	2	56.20%	11.90%		
		8	94.00%	12.90%		
		16	96.10%	11.80%		
		32	96.40%	8.40%		
		256	96.40%	5.20%		

TABLE I: Results on MNIST, Fashion-MNIST, CIFAR-10, GTSRB and ImageNet with color-depth reduction. k is the number of bins. $k = 256$ means we use all possible codes in the input space, that is we don't discretize images.

- *Representativity*. There exists a classifier F that has good accuracy on the discretized data based on $\mathcal{C}$, as described by the framework above.

Therefore, one may want to apply common clustering algorithms, such as k-means and k-medoids, to find such separable codes. Note however, that these algorithms do not make a guarantee of separability. In search for separability, we therefore try both k-medoids and another algorithm that we develop based on density estimation. We next describe the two algorithms.

1) Density Estimation-based Algorithm: We devise a new algorithm to construct separable codes based on density estimation and greedy selection on all the pixels $\mathcal{P}$ in the training data. This algorithm is described in Algorithm 1. This algorithm takes as input a set of images $\mathcal{D}$, a kernel function for density estimation, and number of codes k and a distance parameter r. It repeats k times and at each time, first estimates the densities for all pixel values, then adds the one with highest density to the codebook, and finally, remove all pixels within r distance of the picked.

Algorithm 1 DERIVING CODES VIA DENSITY ESTIMATION

Input: A training dataset $\mathcal{D}$, distance parameter r and number of codewords k, a kernel function $\rho(\cdot, \cdot)$.

Output: a set of codewords $\mathcal{C} = \{c_1, c_2, \cdots, c_k\}$.

1: Let $\mathcal{P}$ denote all the pixels from images in $\mathcal{D}$.
2: **for** $i = 1, 2, \cdots, k$ **do**
3: For each pixel value v, estimate its (unnormalized) density as $h[v] = \sum_{p \in \mathcal{P}} \rho(p, v)$.
4: Set $c_i \leftarrow \arg\max_v h[v]$, and $\mathcal{P} \leftarrow \mathcal{P} - \{p \in \mathcal{P} : \|p - c_i\| \leq r\}$.
5: **end for**

Instantiation. There are many possible kernel functions we can use to instantiate Algorithm 1. In this work we use the simplest choice, the identity kernel, $\rho(p_1, p_2) = 1$ if $p_1 = p_2$ and 0 otherwise. In that case, the density estimation at line 3 above becomes counting the frequencies of a pixel v in $\mathcal{P}$. The values of k and r can be tuned and we will report results for different choices of these parameters.

Algorithm 2 DERIVING CODES VIA k-MEDOIDS

Input: A training dataset $\mathcal{D}$, number of codewords k, number of iterations T, a distance function $d(\cdot, \cdot)$.

Output: a set of codewords $\mathcal{C} = \{c_1, c_2, \cdots, c_k\}$.

Let $\mathcal{P}$ denote all the pixels from images in $\mathcal{D}$.

For any set of codewords $\mathcal{C}$, define the k-medoid cost (w.r.t. to the distance function d) as

$$\text{cost}(\mathcal{P}, \mathcal{C}) = \sum_{p \in \mathcal{P}} \min_{c' \in \mathcal{C}} d(p, c').$$

Randomly pick k pixels as the initial medoids $\mathcal{C} = \{c_1, c_2, \cdots, c_k\}$.

for $t = 1, 2, \cdots, T$ **do**
  for Each pixel $p \notin \mathcal{C}$ and each $c \in \mathcal{C}$ **do**
    Let $\mathcal{C}_{c,p} = \mathcal{C} \setminus \{c\} \cup \{p\}$
    if $\text{cost}(\mathcal{P}, \mathcal{C}_{c,p}) < \text{cost}(\mathcal{P}, \mathcal{C})$ **then**
      Set $\mathcal{C} \leftarrow \mathcal{C}_{c,p}$
    end if
  end for
end for

2) k-Medoids Algorithm: The k-medoids algorithm is similar to the k-means algorithm and aims to minimize the distance between points inside a cluster to the cluster centroid, a point in the cluster, which is representative of the entire cluster. The parameter k is an input to the algorithm (a hyperparameter). The algorithm works as follows: we initially select k points from the given data points as the medoids. Each point is then associated with the closest medoid. The cost of the association is calculated as the sum of pairwise distance between each point and its medoid. The algorithm then iteratively continues by switching a medoid with another data point and checking if this switch of medoids reduces the cost. See Algorithm 2 for the details.

Typically, the ℓ_1 distance, also called the Manhattan distance, is used as the distance metric, and is also the distance used in our experiments. If ℓ_2 distance (the Euclidean distance) is used, the algorithm reduces to the commonly named k-median algorithm. A third option is ℓ_∞ distance. Experimental results show that using ℓ_2 and ℓ_∞ distances lead to similar performance as ℓ_1, so we only report those results for ℓ_1.

Finally, one can use the popular k-means algorithm for constructing the codewords. However, it is known to be sensitive to outliers, and indeed leads to worse performance in our experiments. So we do not include experimental results using the k-means algorithm.

We now present experimental results using the algorithms using density estimation and k-medoids.

B. Experimental Results

We now present our evaluation of data-specific discretization. Since these approaches are more sophisticated than simple color-depth reduction, it is possible for these approaches to yield robust behavior for a wide range of datasets. To summarize this section, however, our key findings are negative: data-specific discretization techniques do not provide much robustness to complex datasets such as CIFAR-10, GTSRB and ImageNet.

Our experimental setup is similar to that of the previous experiments. We use the same differentiable approximation to T, with c_i this time being the codewords derived from our algorithms and performing operations on pixel space. Our experiments consist of the following three parts:

- We would like to evaluate the effectiveness of the specifically designed code construction Algorithm 1 on different datasets. We would also like to check the variation of robustness as we change the adversarial budget ε.
- As in the color-depth reduction experiments, we would also like to understand how the preprocessing technique provides robustness on different datasets quantitatively as we vary the number of codewords.
- Finally, we would like to see how the k-Medoid algorithm for codebook construction affects the performance.

Effectivenss. We now evaluate the robustness obtained by pixel discretization using the codebook constructed by Algorithm 1 on different datasets. We consider six settings:

1) nat_pre: no defenses, model naturally trained on original data;
2) adv_pre: no defenses, model adversarially trained on original data;
3) disc_nat_pre: discretization defense + model naturally trained on original data;
4) disc_adv_pre: discretization defense + model adversarially trained on original data;
5) disc_nat_re: discretization defense + model naturally trained on preprocessed data;
6) disc_adv_re: discretization defense + model adversarially trained on preprocessed data.

We further vary the adversarial budget ε to give a more complete evaluation of the method.

Figure 1, 2, and 3 show the results on the five datasets, with varying ε. We observe that on MNIST and Fashion-MNIST, the method improves over the naturally trained model significantly, and further improves the adversarially trained models. The improvements with discretization are however still minor for datasets other than MNIST and Fashion-MNIST.

We also confirm the so-called adversarial generalization gap phenomenon previously reported in [22], that is, there is a big gap between training and test robustness accuracy. This suggests there is not sufficient data for improving the robustness. The right subfigure in Figure 3 shows this gap on CIFAR-10.

In general, the results suggest that for complex datasets (such as CIFAR-10, GTSRB, and ImageNet), it is difficult to achieve robustness via pixel discretization. This is potentially because the separability and representativity conditions of codewords cannot be satisfied simultaneously due to the data lacking good structure and the base model having peculiar properties. In Sections V, we will study in detail the failure of the method.

Effect of Number of Codewords. Given the results, one may wonder whether the performance can be improved by tuning the

Dataset	Base Model	k	r	Pre-trained		Re-trained	
				Accuracy	Robustness	Accuracy	Robustness
MNIST	Naturally Trained Model	2	0.6	98.81%	75.35%	99.15%	80.91%
		256	N/A	99.17%	0.00%	N/A	
	Adversarially Trained Model	2	0.6	98.17%	97.24%	98.56%	92.80%
		256	N/A	98.40%	92.72%	N/A	
Fashion-MNIST	Naturally Trained Model	2	0.6	98.81%	75.35%	98.93%	79.93%
		256	N/A	99.10%	0.00%	N/A	
	Adversarially Trained Model	2	0.6	98.17%	97.24%	98.56%	92.55%
		256	N/A	98.53%	87.38%	N/A	
CIFAR-10	Naturally Trained Model	2	64	24.20%	15.90%	68.50%	46.70%
		5	64	39.60%	18.70%	79.90%	46.00%
		10	64	42.70%	16.40%	81.60%	43.40%
		50	48	67.70%	12.50%	89.70%	37.10%
		100	16	86.30%	7.60%	92.80%	23.90%
		300	16	91.30%	7.60%	94.40%	20.50%
		500	16	91.21%	9.01%	93.53%	19.81%
		256^3	N/A	95.01%	4.20%	N/A	
	Adversarially Trained Model	2	64	63.20%	24.40%	68.30%	33.20%
		5	64	76.10%	33.40%	78.70%	40.20%
		10	64	78.30%	34.70%	81.00%	43.80%
		50	48	84.30%	39.50%	84.80%	43.80%
		100	16	86.00%	42.50%	85.40%	44.40%
		300	16	87.30%	46.90%	86.80%	46.60%
		500	16	86.99%	46.77%	87.60%	46.84%
		256^3	N/A	87.25%	45.50%	N/A	
GTSRB	Naturally Trained Model	2	64	44.02%	23.94%	55.39%	29.46%
		5	64	66.65%	21.11%	80.55%	25.42%
		10	64	80.51%	27.80%	84.75%	23.87%
		50	48	90.01%	26.42%	91.48%	23.12%
		100	16	96.62%	16.45%	96.45%	17.91%
		300	16	96.80%	16.22%	96.84%	17.08%
		500	16	96.82%	15.49%	96.80%	16.25%
		256^3	N/A	97.35%	7.83%	N/A	
	Adversarially Trained Model	2	64	54.79%	43.18%	58.53%	43.94%
		5	64	74.56%	53.88%	81.43%	58.52%
		10	64	85.83%	66.84%	85.93%	67.36%
		50	48	91.09%	74.26%	91.34%	74.48%
		100	16	93.97%	74.74%	94.72%	76.06%
		300	16	93.96%	75.31%	94.78%	76.18%
		500	16	93.98%	75.54%	94.73%	76.66%
		256^3	N/A	94.11%	74.34%	N/A	
ImageNet	Naturally Trained Model	10	64	53.90%	23.00%	N/A	
		50	48	77.10%	28.50%		
		100	16	88.40%	32.40%		
		200	16	89.50%	32.80%		
		300	16	89.30%	34.80%		
		500	16	92.10%	34.10%		
		256^3	N/A	94.50%	27.50%		
	Adversarially Trained Model	10	64	62.60%	7.50%		
		50	48	85.30%	11.20%		
		100	16	92.80%	12.00%		
		200	16	94.00%	13.30%		
		300	16	94.20%	14.10%		
		500	16	94.80%	16.00%		
		256^3	N/A	96.40%	5.20%		

TABLE II: Results on MNIST, Fashion-MNIST, CIFAR-10, GTSRB and ImageNet with data-specific pixel discretization. We derive codes via density estimation. k and r are hyper-parameters used to find the codes. $k = 256$ in MNIST and Fashion-MNIST and $k = 256^3$ in CIFAR-10, GTSRB and ImageNet mean we use all possible codes in the input space, that is we don't discretize images.

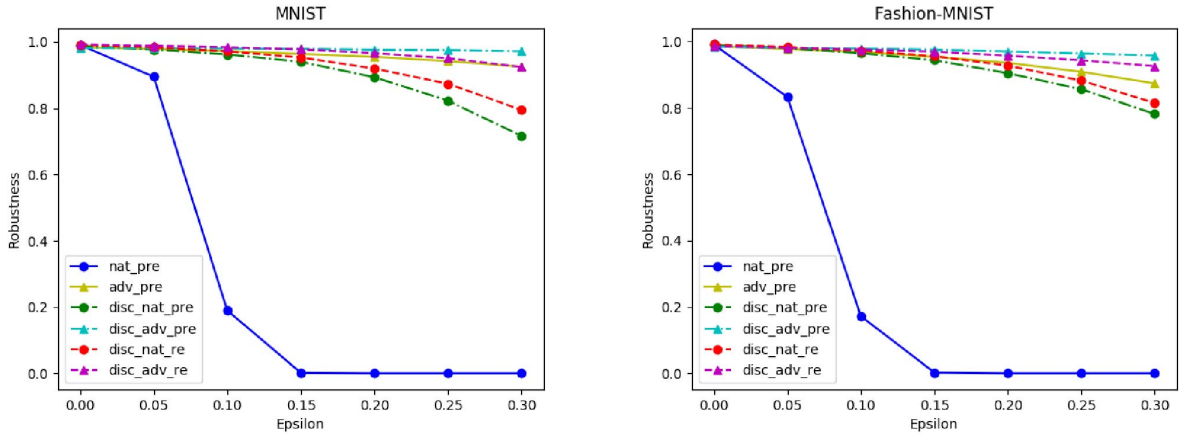
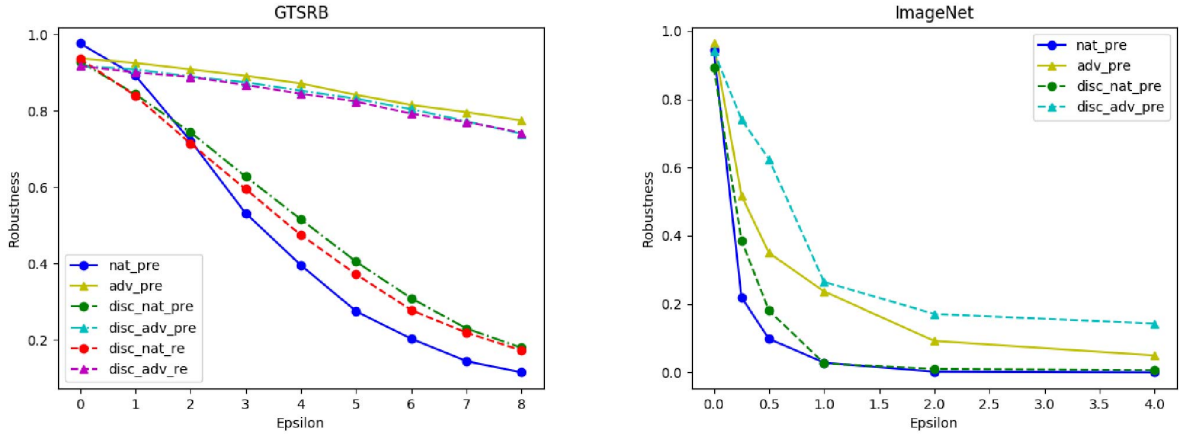


Fig. 1: Results on MNIST and Fashion-MNIST using 2 codes under 100 steps attack. The method significantly improves robustness on naturally pre-trained model and adversarially pre-trained model.



(a) (b)

Fig. 2: Results on GTSRB and ImageNet. (a) Results on GTSRB using 50 codes under 40 steps attack. On the naturally trained model, our method could improve its robustness. However, there is little improvement in robustness on the adversarially pretrained model. (b) The results on ImageNet using 300 codes under 10 steps attack. The method slightly improves robustness on either naturally pre-trained models or adversarially pre-trained models.

parameters of the code construction Algorithm 1, in particular, the number of codewords k. Here, we study the relationship between number of codewords and accuracy and robustness. Table II shows the results on MNIST, Fashion-MNIST, CIFAR-10, GTSRB and ImageNet dataset. For MNIST and Fashion-MNIST, we only report results using 2 codewords since we can achieve very good accuracy and robustness using 2 codewords. For other dataset, we report results on a range of k.

From the results, we can know without attacks, high accuracies can be achieved with only a few number of codewords (e.g., 100), especially when models are retrained. On naturally trained models, with fewer codewords, we gain more robustness. This is because the distances between the codewords are larger and it is harder for the attacker to change the discretization results. On adversarially trained models, an increasing number of codewords leads to better robustness. This is different from the naturally trained case, potentially because the the data points are further away from the decision boundary of adversarially trained models, and thus increasing number of codes does not make it easier for the attacker to change the discretization outcome while giving more representativity

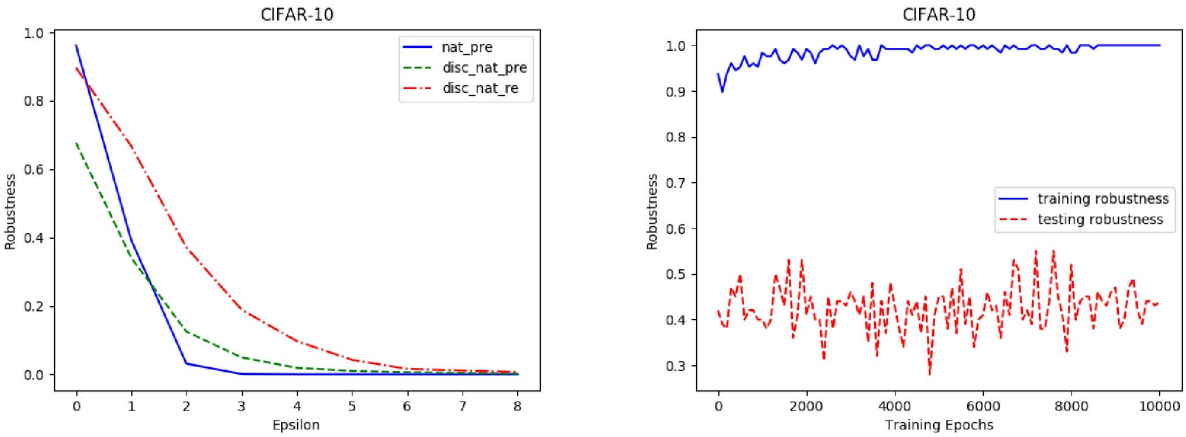


Fig. 3: Results on CIFAR-10 using 50 codes under 40 steps attack. (a) On the naturally trained model, the method could improve its robustness. On the adversarially trained model, there is little improvement so we omit the plot. Instead, we plot the adversarial gap underlying this failure. (b) Adversarial gap phenomenon on CIFAR-10. When we adversarially retrain the model on discretized images, we can achieve nearly 100% robustness in training but no higher than 50% in test. This is referred to as an adversarial generalization gap [22].

Dataset	Base Model	k	Pre-trained		Re-trained	
			Accuracy	Robustness	Accuracy	Robustness
MNIST	Naturally Trained Model	2	98.81%	75.43%	98.99%	80.66%
	Adversarially Trained Model	2	98.17%	97.20%	98.50%	92.20%
Fashion-MNIST	Naturally Trained Mode	2	98.85%	76.42%	99.00%	81.23%
	Adversarially Trained Model	2	98.14%	97.21%	98.52%	92.72%
CIFAR-10	Naturally Trained Model	300	92.73%	8.56%	93.91%	17.08%
	Adversarially Trained Model	300	86.97%	46.68%	87.53%	47.11%
GTSRB	Naturally Trained Model	300	97.31%	14.61%	97.39%	15.02%
	Adversarially Trained Model	300	93.98%	75.56%	95.06%	76.46%
ImageNet	Naturally Trained Model	300	92.30%	33.80%	N/A	
	Adversarially Trained Model	300	94.90%	12.20%		

TABLE III: Results on MNIST, Fashion-MNIST, CIFAR-10, GTSRB and ImageNet with data-specific pixel discretization. We derive codes via k-medoids. k is the number of medoids.

leading to better results. This also means that discretization does not lead to significantly change in the robustness. This is confirmed by observing that adversarially trained models without discretization can get 45.60% robustness on CIFAR-10. This is further investigated in Section V.

Using k-Medoids. Finally, we also try k-medoids algorithm for generating the codebook. As mentioned, using ℓ_2 or ℓ_∞ distances leads to similar performance as ℓ_1, so we report only the results for ℓ_1 in Table III. As is quite clear from the table, the results are on par with the results when using the density estimation algorithm. With MNIST and Fashion-MNIST, this is somewhat obvious as both algorithms select codewords at or close to 0 and 1. With other dataset, while not so obvious, the accuracy and robustness still remain similar to those with the codebook construction algorithm via density estimation.

V. Analysis of Results

In this section we address the following question: *when does pixel discretization work?* Experimental results show that the pixel discretization defense methods can achieve strong performance on the MNIST data set and alike, but fail on more complex data sets, such as CIFAR-10 and ImageNet. A better understanding of the conditions under which the simple framework will succeed can help us identify scenarios where our technique is applicable, and also provide insights on how to design better defense methods.

We aim at obtaining better insights into its success by the pursuing two lines of studies. First, we propose a theoretical model of the distribution of the pixels in the images and then prove that in this model the pixel discretization provably works. Second, inspired by our theoretical analysis, we propose a method to compute a certificate for the performance of the defense method on given images and a given adversarial budget.

This certificate is the lower bound of the performance for *any* attacks using a given adversarial budget. If the certified robust accuracy is high, then it means that the defense is successful for any attack within the budget, not just for existing. This is much desired in practice, especially considering that new and stronger attacks are frequently recently proposed for DNNs. The certificate, as a lower bound, also allows a rough estimation of the performance, when combined with the robust accuracy under the currently available attacks which is an upper bound.

1) An idealized model and its analysis: To garner additional insights, we propose and analyze an idealized model under which we can precisely analyze when the adversarial robustness can be improved by pixel discretization method using codewords constructed by Algorithm 1.

In the center of our analysis is a generative model for images, i.e., a probabilistic model of the distributions of the images. Roughly speaking, in this model, we assumes that there exists some "ground-truth" codewords that are well separated, and the pixels of the images are slight perturbations of these codewords, and thus form well separated clusters. This idealized model of the images are directly inspired by the known clustering structure of the pixels in the MNIST data set, i.e., most pixels from MNIST are either close to 1 or close to 0. Given such a clustering structure, the codeword construction algorithm (Algorithm 1) can find codewords that are good approximations of the ground-truth codewords, and the adversarial attacks cannot change the discretization results much. In summary, our analysis results suggest the following: Suppose that data is good in the sense that it can be "generated" using some "ground-truth" codewords that are sufficiently well separated; then, as long as we can find a ν-approximation for the ground truth codewords and we have a base model F that is robust with respect to the ν-budget, it follows $F(T(\cdot))$ is immune to any adversarial attack with a small budget $\varepsilon \leq 5\nu$, thus providing a boost of adversarial robustness. Or in other words, pre-processing provides a 5x boost on adversarial robustness. We now present details.

An idealized generative model of images. Each image $\mathbf{x}$ is viewed as a d-dimensional array (a typical image of width W and height H can be flattened into an array of dimension $W \times H$). Suppose each pixel $\mathbf{x}[i]$ is 3-dimensional vector of discrete values in $[K]$. Assume that there is a set of ground-truth codewords $\mathcal{C}^* = \{c_1^*, \ldots, c_k^*\}$, where the codewords $c_i^* \in [K]^3$ lies in the same space as the pixels. Also, assume that the codewords are well separated so that $\|c_i^* - c_j^*\| \geq \Gamma$ for some large Γ.

Now we specify the generative process of an image. Each image is generated in two steps, by first generating a "skeleton image" $\mathbf{u}$ where each pixel is a codeword, and then adding noise to the skeleton. We do not make assumptions about the distribution of the label y. Formally,

1) $\mathbf{u}$ is generated from some distribution over $(\mathcal{C}^*)^d$, where the marginal distributions satisfy
$$\frac{1}{d}\sum_{j\in[d]} \Pr(u_j = c_i^*) = \frac{1}{k}, \forall i \in [k].$$
2) $\mathbf{x}[i] = \mathbf{u}[i] + \zeta_i$, where ζ_i takes valid discrete values so that $\mathbf{x}[i] \in [K]^3$, and
$$\|\zeta_i\| \leq \Gamma/8, \text{ and } \Pr[\|\zeta_i\| = t] = \alpha \exp(-t^2/\sigma^2)$$
where t takes valid discrete values, σ is a parameter, and α is a normalization factor.

We would like to make a few comments about our generative model. First, the assumption on the skeleton $\mathbf{u}$ is very mild, since the only requirement is that the probability of seeing any codeword is the same $1/k$, i.e., randomly pick a pixel in $\mathbf{u}$ and it is equal to any codeword with the same probability. This is to make sure that we have enough pixels coming from any codeword in the training data, so this condition can be relaxed to $\frac{1}{d}\sum_{j\in[d]} \Pr(u_j = c_i^*) \geq \varepsilon_c$ for a small ε_c. We set $\varepsilon_c = 1/k$ for simplicity of the presentation.

For the second step of the generative process, the assumptions that the noise ζ_i takes discrete values and that $\|\zeta_i\| \leq \Gamma/8$ are also for simplicity. The actually needed assumption is that with high probability, the noise is small compared to Γ, the separation between the ground-truth codewords.

Quantifying robustness. We now prove our main theoretical result in the idealized model.

As a first step, we first show that the codewords constructed are quite close to the ground-truth. Formally, we say that a set of codewords $\{c_1, \ldots, c_k\}$ is a *ν-approximation* of $\mathcal{C}^*$ if for any i, $\|c_i - c_i^*\| \leq \nu$. For the above generative model, one can show that the codewords found by Algorithm 1 are ν-approximation of the ground truth $\mathcal{C}^*$.

Lemma 1. *Let N denote the number of pixels in the training set. For any $\delta > 0$, if $\Gamma > 16\nu$ where ν is defined as in Proposition 1, then with probability at least $1-\delta$, Algorithm 1 with $r = 2\nu$ outputs a set of codes $\mathcal{C} = \{c_1, \ldots, c_k\}$ such that $\|c_i - c_i^*\| \leq \nu, \forall i$.*

Proof. Let x denote a pixel in the generated images. We have for any i, $\Pr[x = c_i^*] \geq \alpha/k$, and for any c such that $\|c - c_i^*\| \geq \nu$ for all i, $\Pr[x = c] \leq \alpha \exp(-\nu^2/\sigma^2)$. By Hoeffding's inequality [23, Section 2.6], with probability at least $1-\delta$, the empirical estimations satisfy
$$\widehat{\Pr}[x = c_i^*] \geq \alpha/k - \gamma,$$
$$\widehat{\Pr}[x = c] \leq \alpha \exp(-\nu^2/\sigma^2) + \gamma.$$
Then $\widehat{\Pr}[x = c_i^*] > \widehat{\Pr}[x = c]$. Note that $r = 2\nu$ and $\Gamma > 16\nu$, so for each c_i^*, Algorithm 1 picks a code from the neighborhood around c_i^* of radius ν exactly once. This completes the proof. □

Our main result, Proposition 1, then follows from Lemma 1. To state the result, let us call a transformation G to be a *ν-code perturbation* of $\mathcal{C}^*$ if given any skeleton $\mathbf{u}$, $G(\mathbf{u})$ replaces any c_i^* in it with a code c_i' satisfying $\|c_i' - c_i^*\| \leq \nu$. With this definition we show that, on an image attacked with adversarial budget $\varepsilon \leq 5\nu$, our discretization T will output a ν-code perturbation on $\mathbf{u}$. Lemma 1 then leads to the following proposition (proof is straightforward).

Proposition 1. *Assume the idealized generative model, and* $\Gamma > 16\nu$ *where*

$$\nu = \sigma\sqrt{\log\frac{1}{1/k - 2\gamma/\alpha}}, \quad \gamma = \sqrt{\frac{4}{N}\log\frac{K}{\delta}}, \tag{1}$$

where N *is the number of pixels in the training dataset. Assume* $r = 2\nu$ *in Algorithm 1. Then for any* $\varepsilon \leq 5\nu$*,*

$$\begin{aligned}&\Pr_{(\mathbf{x},\mathbf{u},y)}[F(T(\mathbf{z})) = y \text{ for any } \|\mathbf{z}-\mathbf{x}\| \leq \varepsilon]\\ \geq\ &\min_G \Pr_{(\mathbf{x},\mathbf{u},y)}[F(G(\mathbf{u})) = y]\end{aligned}$$

with probability at least $1-\delta$*, where the minimum is taken over all* ν*-code perturbation* G*.*

Essentially, this proposition says that one can "reduce" defending against 5ν adversarial attacks to defending against ν-code perturbations G. Therefore, as long as one has a base model F that is robust to small structured perturbation (i.e., the ν-code perturbation), then one can defend against *any* 5ν adversarial attacks, which is a significant boost of robustness. Indeed, we observed that the intuition is consistent with the experimental results: the method gives better performance using an adversarially trained $F(\cdot)$, than a naturally trained F, on structured data like MNIST.

This analysis also inspires the following simple approach for computing a certificate of the robust accuracy for a set of codewords on given images and a given adversarial budget.

2) Certification for discretization defense: The analysis shows that the method succeeds when the adversarial attack cannot cause significant changes after discretization, and the base model is robust to the slight change in the discretized image. In fact, the intuition is also applicable to general cases beyond the idealized model. In particular, one can empirically check if such conditions are met when the images, the base model, and the adversarial budget are all given. This observation then gives our algorithm for computing the certificate for the defense method.

Now we formally derive the certificates for the defense. Given the codewords $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$ used in the pixel discretization, for a pixel $\mathbf{x}[i]$, let $c^*(\mathbf{x}[i])$ denote its nearest code in $\mathcal{C}$, and define

$$\mathcal{C}(\mathbf{x}[i]) = \{c : \|\mathbf{x}[i] - c\| < \|c^*(\mathbf{x}[i]) - \mathbf{x}[i]\| + 2\varepsilon\} \tag{2}$$

where ε is the adversarial budget.

Then after perturbation δ bounded by ε, the distance between the perturbed pixel $\mathbf{x}[i]+\delta$ and $c^*(\mathbf{x}[i])$ is

$$\begin{aligned}\|\mathbf{x}[i] + \delta - c^*(\mathbf{x}[i])\| &\leq \|\mathbf{x}[i] - c^*(\mathbf{x}[i])\| + \|\delta\|\\ &\leq \|\mathbf{x}[i] - c^*(\mathbf{x}[i])\| + \varepsilon\end{aligned}$$

by the triangle inequality. On the other hand, the distance between the perturbed pixel $\mathbf{x}[i]+\delta$ and any $c \notin \mathcal{C}(\mathbf{x}[i])$ is

$$\begin{aligned}\|\mathbf{x}[i] + \delta - c\| &\geq \|\mathbf{x}[i] - c\| - \|\delta\|\\ &\geq \|\mathbf{x}[i] - c\| - \varepsilon.\end{aligned}$$

By the definition of $\mathcal{C}(\mathbf{x}[i])$, we know that

$$\|\mathbf{x}[i] - c\| - \varepsilon > \|\mathbf{x}[i] - c^*(\mathbf{x}[i])\| + \varepsilon.$$

So after perturbation, the pixel $\mathbf{x}[i]+\delta$ can only be discretized to a code in $\mathcal{C}(\mathbf{x}[i])$.

Then all possible outcomes of the discretization after perturbation are

$$\mathcal{S}(\mathbf{x}) = \{\mathbf{z} = (\mathbf{z}[1], \ldots, \mathbf{z}[d]) : \mathbf{z}[i] \in \mathcal{C}(\mathbf{x}[i])\}. \tag{3}$$

This then leads to the following local certificate for a given image, and global certificate for the whole distribution.

Local certificate. For a data point $(\mathbf{x}, y)$, if for any $\mathbf{z} \in \mathcal{S}(\mathbf{x})$ we have $F(\mathbf{z}) = y$, then it is guaranteed that $\mathbf{x}$ is correctly classified by $F(T(\cdot))$ to y even under the adversarial attack with ε-budget. Formally, let $\mathbb{I}_F(\mathbf{x}, y)$ be the indicator that $F(\mathbf{z}) = y$ for all $\mathbf{z} \in \mathcal{S}(\mathbf{x})$, then

$$\begin{aligned}\mathbb{I}_F(\mathbf{x}, y) &= \mathbb{I}[F(\mathbf{z}) = y, \forall \mathbf{z} \in \mathcal{S}(\mathbf{x})]\\ &\leq \mathbb{I}[F(T(\mathbf{z})) = y \text{ for any } \|\mathbf{z}-\mathbf{x}\| \leq \varepsilon],\end{aligned} \tag{4}$$

so $\mathbb{I}_F(\mathbf{x}, y)$ is a lower bound on the robust accuracy for this data point. It serves as a local certificate for $(\mathbf{x}, y)$.

Global certificate. Define $s = \mathbb{E}_{(\mathbf{x},y)\sim D}[\mathbb{I}_F(\mathbf{x}, y)]$. Then clearly, we have

$$s \leq \Pr_{(\mathbf{x},y)\sim D}[F(T(\mathbf{z})) = y \text{ for any } \|\mathbf{z}-\mathbf{x}\| \leq \varepsilon],$$

so s serves as a lower bound for the robustness accuracy of the defense on the whole data distribution.

Of course, computing the exact value of s is not feasible, as it requires access to the true data distribution. Fortunately, this certificate s can be easily estimated on a validation set of data points. Even with a medium number of samples we can compute an estimation $\hat{s}_*$ that is with high probability a close lower bound of s. Formally, applying a standard concentration bound (in particular, the Hoeffding's inequality) leads to the following.

Proposition 2. *Let* $\hat{s}$ *be the fraction of* $\mathcal{I}_F(\mathbf{x}, y) = 1$ *on a set* $\mathcal{V}$ *of* m *i.i.d. samples from the data distribution. Then with probability at least* $1-\delta$ *over* $\mathcal{V}$*,*

$$\begin{aligned}&\Pr_{(\mathbf{x},y)\sim D}[F(T(\mathbf{z})) = y \text{ for any } \|\mathbf{z}-\mathbf{x}\| \leq \varepsilon]\\ \geq\ &\hat{s}_* := \left(1 - \sqrt{\frac{1}{2m}\log\frac{1}{\delta}}\right)\hat{s}.\end{aligned}$$

Note that computing the certificate needs enumerating $\mathcal{S}(\mathbf{x})$ that can be of exponential size. However, when the pixels are well clustered, most of them are much closer to their nearest code than to the others, and thus will not be discretized to a new code after perturbation, i.e., $|\mathcal{C}(\mathbf{x}[i])| = 1$. Then $\mathcal{S}(\mathbf{x})$ is of small size and the certificate is easy to compute. This is indeed the case on the MNIST data, which allows us to compute the estimated certificate $\hat{s}$ in our experiments.

3) Certifying robustness on datasets: Here we compute the certificate derived above on real world data sets, including MNIST and others. It is expected that the certificate can be computed in reasonable time on well structured data like MNIST, while it may not be efficiently computed or provide no useful lower bound for other less structured data sets.

ε	b	Unable	Success($\hat{s}$)	Fail	$\hat{s}_*$
0.00	0	0.0%	98.1%	1.9%	96.61%
0.05	30	0.43%	97.43%	2.14%	95.95%
0.10	30	1.3%	96.43%	2.27%	94.97%
0.15	26	29.18%	69.01%	1.81%	67.96%
0.20	25	74.38%	24.96%	0.66%	24.58%
0.25	25	89.48%	10.37%	0.15%	10.21%
0.30	25	95.67%	4.31%	0.02%	4.24%

TABLE IV: Certificate results on MNIST for different adversarial budget ε. For computational reasons, we set a threshold b, and if $|\mathcal{S}(\mathbf{x})| > 2^b$ where $\mathcal{S}(\mathbf{x})$ is defined in Eqn (3), we report it as an Unable case. $\hat{s}$ and $\hat{s}_*$ are defined in Proposition 2.

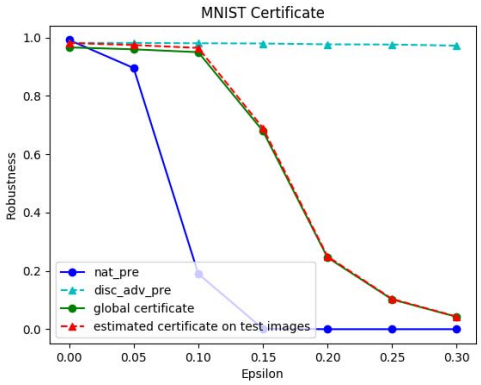


Fig. 4: Certificate results on MNIST for the codewords constructed by Algorithm 1, compared to the empirical result without defense (nat_pre), and the empirical result with defense (disc_adv_pre). We plot two certificate results: estimated certificate on test images $\hat{s}$, and global certificate $\hat{s}_*$, as defined in Proposition 2. Also see Table IV for the detailed numbers.

MNIST Certificate. We compute the estimated certificate ($\hat{s}$ in in Proposition 2) on 10000 test images of MNIST for the codewords constructed by Algorithm 1. We also compute the global certificate ($\hat{s}_*$ in Proposition 2) for the same set of codewords, where the failure probability δ is set to be 0.01. For computational reasons we set up a threshold b and for $\mathbf{x}$ with $|\mathcal{S}(\mathbf{x})| > 2^b$ where $\mathcal{S}(\mathbf{x})$ is defined in (3), we report Unable and treat them as failure cases when computing the certificates.

Our results are provided in Table IV, and exact values of $\hat{s}$ and $\hat{s}_*$ are compared with experimental results in Figure 4. There exists some methods to compute estimated certificate robustness, like [24], [25], [26]. The state-of-art estimated certificate robustness on MNIST under ℓ_∞ perturbations with $\varepsilon = 0.1$ is 94.18% [25], with a fairly sophisticated method. Our discretization defense, which is much simpler and more efficient, gives a better estimated certificate robustness of 96.43%. This demonstrate the effectiveness of this simple certificate method. It also verifies the analysis for our idealized generative model of images, providing positive support for the conclusion that in the presence of well separated clusters of pixels, the pixel defense method on a good base model can successfully defend the adversarial attacks.

A. How hard is a dataset to defend?

The discretization defenses against adversarial attacks are effective only for some datasets (e.g, MNIST). In the last section we analyzed why on datasets like MNIST our method can succeed. But it is also interesting and can help improve the method if we can understand why our techniques fail on other datasets and quantify how far they are to the conditions for the success. In this section, we propose metrics for these purposes and provide empirical results using these metrics.

Before formally defining these metrics, let us first describe the concrete context for using these metrics to show the level of hardness for defense. These include the intuition about the base model, and our goal for using the metrics.

First, let F denote our base model that is *naturally trained*. The fact that they fail miserably under adversarial attacks with small adversarial budgets suggests that such models have "peculiar behaviors". Concretely, for any input feature vector $\mathbf{x}$, we define its robust radius with respect to F and ℓ_p norm, $\delta_{F,p}(\mathbf{x})$, as

$$\delta_{F,p}(\mathbf{x}) = \sup_{\delta}\{\forall\, \mathbf{x}', \| \mathbf{x}' - \mathbf{x} \|_p \leq \delta, F(\mathbf{x}') = F(\mathbf{x})\}.$$

In other words, $\delta_{F,p}(\mathbf{x})$ is the largest radius (in ℓ_p-norm), under which F still gives consistent predictions around $\mathbf{x}$. The fact that the base model is vulnerable to small adversarial perturbations can then be formalized as the following statement:

Vulnerability of natural models: *For most points $\mathbf{x}$ in the domain, $\delta_{F,p}(\mathbf{x})$ is small ("infinitesimal").*

Now let T denote the discretization preprocessing algorithm, which takes as input $\mathbf{x}$, and outputs another feature vector $\mathbf{z} = T(\mathbf{x})$ (of the same dimensionality), and the final classification is $F(\mathbf{z}) = F(T(\mathbf{x}))$. Our goal is the following:

Goal: *Arguing that it will be very hard to come up with a T that works for the naturally trained F, except for trivial situations.*

The intuition is that the base model F is peculiar while the discretization is "regular" (i.e., the discretized image is a product of the same discretization scheme on each of its pixel). The restriction on the discretization, when facing the peculiarity of the base model, prevents simultaneously getting a good accuracy and getting consistent output in the neighborhood of the input data.

To get the intuition, consider the following intentionally simplified setting: each image has only one pixel taking values in $[0, 1]$. Divide $[0, 1]$ into m many intervals $[(i-1)/m, i/m]$ for $1 \leq i \leq m$, and the images in the i-th interval belong to class 0 if i is even, and belong to class 1 if i is odd. Now discretize with k codewords. When m is much larger than k, no matter where we place the codewords, the accuracy will not be good: if I_c denotes the set of images discretized to the codeword c, then we know that many intervals with different

classes will fall into the same I_c's, resulting in bad accuracy for any classifiers on top of the discretized images. On the other hand, if k is comparable to m, the distance between the codewords will be small and the attacker can easily change the discretized outcome by perturbing the image from one I_c to another $I_{c'}$. Therefore, the fracture decision boundary of the data and the regularity of the discretization combined together can prevent obtaining at the same time good accuracy and unchanged predictions w.r.t. adversarial perturbations.

Now let us provide a more detail argument below.

1) An argument based on equivalence classes: We think of T as creating equivalence classes, where $\mathbf{x} \sim_T \mathbf{x}'$ if $T(\mathbf{x}) = T(\mathbf{x}')$, that is, they are discretized to the same output. Now, given an ℓ_p-norm adversary with adversarial budget ε, for each input $\mathbf{x}$, we can consider the set of equivalence classes obtained in the ball $N_p(\mathbf{x}, \varepsilon)$. In mathematical terms this is exactly:

$$\overline{N_p}(\mathbf{x}, \varepsilon) = N_p(\mathbf{x}, \varepsilon)/\sim_T$$

where $\sim_T$ denotes the equivalence relation induced by T.

Now, we want to argue the following:

1) For a naturally trained model F, $N_p(\mathbf{x}, \varepsilon)$ is tiled with infinitesimal balls of different classification labels given by the base model.
2) In order to keep accuracy, we tend to create small equivalence classes with $\sim_T$. In other words, $|\overline{N_p}(\mathbf{x}, \varepsilon)|$ is large (i.e. there are lot of equivalence classes in a neighborhood around $\mathbf{x}$). In fact, not only we can assume that it is large, but we can assume that $\overline{N_p}(\mathbf{x}, \varepsilon)$ *subtracting the equivalence class of* $\mathbf{x}$ *has a large volume (under uniform measure say).*
3) If the above two hold, then it is now very likely that there is a small ball, with a different label, that lies in a different equivalence class than $\mathbf{x}$. In other words, if we denote $\overline{\mathbf{x}}$ as the equivalence class of $\mathbf{x}$ in $N_p(\mathbf{x}, \varepsilon)/\sim_T$, then that there is $\mathbf{x}'$ such that $\overline{\mathbf{x}} \neq \overline{\mathbf{x}'}$, and $F(\mathbf{x}) \neq F(\mathbf{x}')$, and we can easily find an adversarial example.

2) An instantiation for ℓ_∞ norm: We note that for ℓ_∞ norm, it is very easy to construct $\overline{N_\infty}(\mathbf{x}, \varepsilon)$. Specifically, suppose that the input feature vectors are d-dimensional. Then for each "pixel" $\mathbf{x}[i]$, we can consider the discretization of $\mathbf{x}[i]$ under T, i.e., $T(\mathbf{x}[i])$ (note that we are abusing the notation and now consider the effect of T on a single pixel). So for every i, we can consider the equivalence classes created at dimension i for the interval: $C_i = \{T(z) \mid \|z - \mathbf{x}[i]\| \leq \varepsilon\}$. Then all the equivalence classes we can create for an image $\mathbf{x}$ are simply: $\prod_{i=1}^{d} C_i$. Therefore, the $|\overline{N_\infty}(\mathbf{x}, \varepsilon)|$ becomes $\prod_{i=1}^{d} |C_i|$. We can thus either use this number as a measure of how fragmented the ball is under T, reflecting how difficult it is to do the defense.

We would like to measure the quantity $\prod_{i=1}^{d} |C_i|$ for the datasets we use. In Figures 5a and 5b, we plot the CDF (cumulative distribution function) of $\frac{1}{d} \log_k \prod_{i=1}^{d} |C_i|$ for both MNIST and CIFAR-10, where k is the number of codes (2 for MNIST and 300 for CIFAR-10 and d is the number of pixels in one image (28^2 for MNIST and 32^2 for CIFAR-10).

For MNIST, the measure of the median figure is about 0.06, which means the median $\prod_{i=1}^{d} |C_i|$ is approximately 47. Out of 784 pixels of an image, in the median case only 47 can be perturbed to a different equivalence class. This shows why it is easy to achieve robustness on MNIST. For CIFAR-10, the measure for median figure is about 0.27, which implies that the number of equivalence classes $\prod_{i=1}^{d} |C_i|$ is 300^{276}, which is a huge number and supports the hypothesis that defending CIFAR-10 will be hard using pixel discretization techniques.

Furthermore, this equivalence class argument also gives an explanation to the adversarial gap phenomenon (see Figure 3): adversarial training only helps to adjust the labels of the equivalence classes of the perturbations of the training data points, which could be quite different from those classes of the perturbations of the test data points due to the peculiar decision boundary. This then leads to high robust accuracy in training time but low in test time.

B. How Separable is a Dataset?

One key requirement for pixel discretization is separability. In a separable dataset, pixel clusters are far from each other such that perturbation of a pixel in one cluster cannot make the resultant pixel to move to another cluster. This is directly related to the equivalence classes argument discussed above.

We visualize MNIST and CIFAR-10 datasets to study if they can have separable clusters. Figure 6 presents visualizations of CIFAR-10 pixel neighborhoods. Each axis in the plots corresponds to a color channel. There are about 4 million distinct pixels in the CIFAR-10 training dataset. These being too many pixels, we plot only samples of these pixels in this figure. Figure 6a shows 40,000 pixels on the plot, with each axis representing a color channel. We assume a colorscale range of $[0, 1]$. The color of each pixel x is given by $|\mathcal{N}_\infty(x, \varepsilon)|/\max_x |\mathcal{N}_\infty(x, \varepsilon)|$, where $\mathcal{N}_\infty(x, \varepsilon)$ is the ℓ_∞ neighborhoods of radius ε around x. For these figures we use $\varepsilon = 8$. Note that the maximum neighborhood size is over 1.5 million and there are many pixels with neighborhood sizes in hundreds or a few thousands and hence most of pixels have colors close to zero.

To overcome the long-tail distribution of neighborhood sizes, we also do a log scale plot in Figure 6b, this time using a sample of 400,000 pixels and using $\log |\mathcal{N}_\infty(x, \varepsilon)|/\log \max_x |\mathcal{N}_\infty(x, \varepsilon)|$ to color the pixels on a colorscale of $[0, 1]$. As can be seen from both these plots, *there does not appear to be a clear, separable clustering in the CIFAR-10 dataset.* Note that the line $x = y = z$, which corresponds to gray colors on RGB, is lighter-colored, implying large neighborhoods in this area. In fact, we have verified that our density estimation-based algorithm selects codewords along this line. Clearly, these codewords are neither very representative of pixels far from this line nor do they lead to the separability property for the clusters around them.

We also visualize the MNIST pixels. Figure 7 presents a histogram of MNIST pixels. As we can see, most of the pixels are black or highly white and very few are in between. This leads to *very good separability* as there are very few pixels

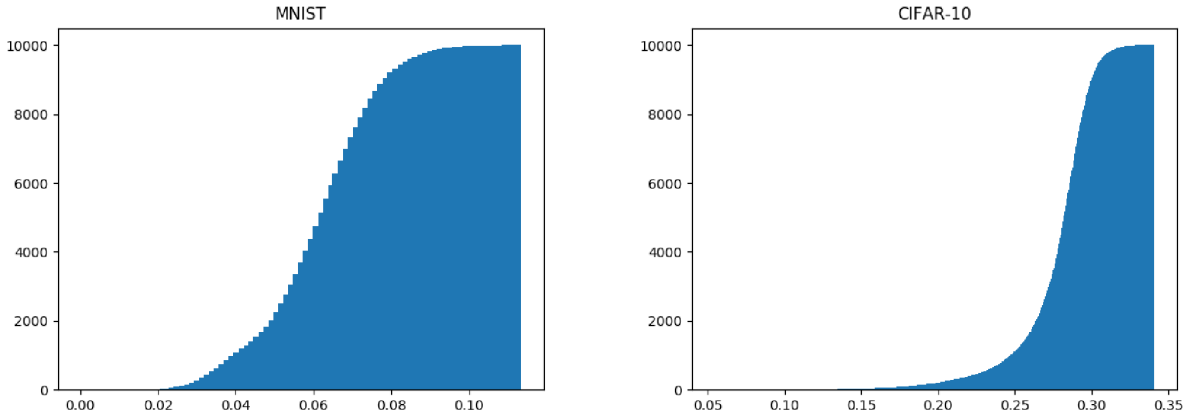
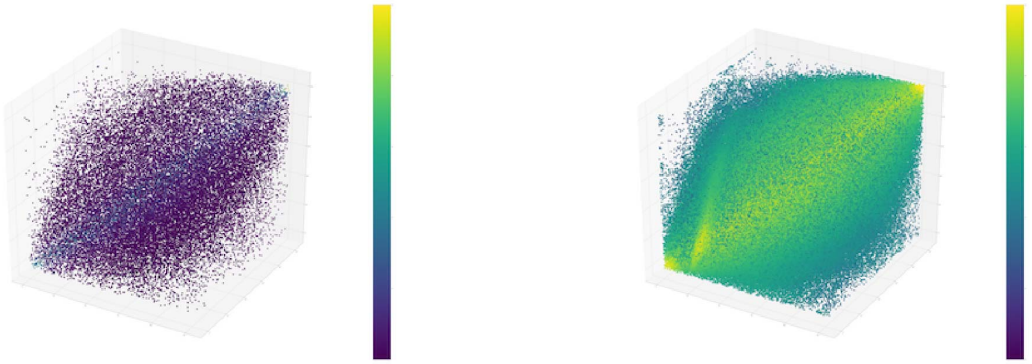


Fig. 5: CDFs measuring the hardness of defending images in MNIST and CIFAR-10.



(a) Linear scale

(b) Log scale

Fig. 6: 3D visualization of CIFAR-10 pixels colored by their neighborhood size. (a) A sample of 40,000 pixels color-coded by their neighborhood sizes in the dataset. (b) A sample of 400,000 pixels color-coded by their neighborhood sizes on logscale. We normalize the colorscale of the two figures using maximum neighborhood size and log of maximum neighborhood size respectively. The maximum neighborhood size is 1,567,080.

that can actually be perturbed to another cluster or equivalence class.

VI. Comments

In the previous section, we developed a theory for evaluating pixel discretization defenses and presented empirical results on CIFAR-10 showing why it is unlikely for pixel discretization techniques to successfully defend against adversarial perturbations on this dataset. While our argument until now was applied on color-depth reduction and the data-specific discretization in Section IV, the same argument can be applied on other discretizations such as the thermometer encoding defense by Buckman et al. [6].

The thermometer encoding defense encodes each pixel value $\mathbf{x}[i, j, l]$ as an L-dimensional vector whose k-th component is 1 if $\mathbf{x}[i, j, l] > k/L$ and 0 otherwise. Thermometer encoding essentially rounds a pixel to one of L levels like color-depth reduction but provides a fancier encoding, which was intended to break gradient descent. Therefore, the equivalence classes argument that we developed in Section V-A can be directly applied here. As our results in Figure 5 have shown, CIFAR-10 will be quite difficult to defend with this technique (the exact statistics on the number of equivalence classes will differ in

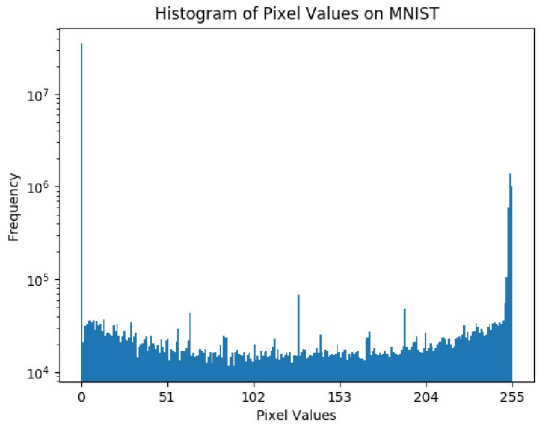


Fig. 7: A histogram of pixel values in MNIST. Note that y-axis is in log scale.

this technique and will also depend on L).

We also observe that most preprocessing defenses to date have been developed in an ad hoc manner, with the only evaluation being how well the defense works against currently-known adversarial attacks. A better lower-bound performance of a preprocessing defense may be obtained by quantifying how much the preprocessing technique results in the reduction of equivalence classes for an image for a given dataset. For pixel discretization, this quantification is somewhat easy as the equivalence classes at the pixel level can be lifted to equivalence classes at the image level. For other preprocessing techniques, the specifics of those techniques will dictate how this quantification is done.

In general, few preprocessing techniques may provide robustness without significantly affecting accuracy to a naturally trained model. A preprocessor T is useful only if it increases the robustness radius (Section V-A) so that $\delta_{F(T(\cdot)),p}(\mathbf{x}) = \delta_{F,p}(T(\mathbf{x})) > \delta_{F,p}(\mathbf{x})$. However, this requirement may result in sacrificing some natural examples that have a label different from $\mathbf{x}$ but are nonetheless mapped by T to $\mathbf{x}$, resulting in a sacrifice of accuracy. This is similar to the equivalence class argument in the previous section. Therefore, the design of the preprocessing defense has to be conservative so as to not sacrifice accuracy too much. A stark example of this is in the accuracy-robustness results of pixel discretization on CIFAR-10 and a naturally-trained model in Table II. Formalizing this intuition for broader class of preprocessing defense methods is left for future work.

VII. Related Work

a) Adversarial settings in machine learning: A number of adversarial settings exist in machine learning. The primary ones are training-time attacks, model inversion, model extraction, and test-time attacks. Training time attacks poison the training data resulting in learning a wrong machine learning model. The attacker may skew the classification boundary to their favor by introducing bad data labeled as good. Data pollution attacks have been studied long before other attacks became relevant [27]. In a model inversion attack, the attacker learns information about data used to train the machine learning model [28], [29], [30]. A similar but stronger setting is the membership inference attack where the attacker identifies whether an individual's information was present in the training data [31]. Model extraction attacks attempt to steal a model simply through blackbox queries [32]. All these settings are different from the setting that this paper focuses on, namely, small perturbations to natural inputs to get them classified differently than original inputs.

b) Adversarial perturbations: Adversarial perturbation attacks usually work by starting with a natural example and solving an optimization problem to derive an adversarial example that has a different label. A number of techniques with varying settings and efficacy have been developed [33], [1], [34], [35], [12], [15]. The BPDA attack [10] is helpful when the gradient of a part of the model is not available so that gradient descent is not possible. The attack overcomes the problem by using the gradient of a differentiable approximation of the function whose gradient is not available. We use this attack as the pixel discretization defense is not differentiable.

c) Defenses against adversarial perturbations: Most defenses can be divided into adversarial training defenses and preprocessing defenses. Currently, the most successful way to defend against adversarial attacks is adversarial training, which trains the model on adversarial examples. This training in general imparts adversarial robustness to the model. The state-of-the-art in this area is Madry et al. [15].

Several other defenses fall under the category of preprocessing defenses. These defenses are model agnostic, not requiring changing the model, and rather simply transform the input in the home of increasing adversarial robustness. We have already discussed pixel discretization works of Xu et al. [5] and Buckman et al. [6]. Other techniques include JPEG compression [2], total variance minimization [3], image quilting [3], image re-scaling [7], and neural-based transformations [8], [9], [4]. Total variance minimization randomly selects a small set of pixels from the input and then constructs an image consistent with this set of pixels while minimizing a cost function representing the total variance in the image. The idea is that adversarial perturbations are small and will be lost when the image is simplified in this way. Similar ideas are behind JPEG compression. Image quilting on the other hand uses a database of clean image patches and replaces patches in adversarial images with the nearest clean patches. Since the clean patches are not adversarial, the hope is that this approach will undo any adversarial perturbation. Image rescaling and cropping can change the adversarial perturbations' spatial position, which is important for their effectiveness. Neural-based approaches train a neural network to "reform" the adversarial examples.

Most of these techniques above have been explicitly broken by the BPDA attack and others are also believed broken under BPDA attack. We also presented compelling evidence why pixel discretization cannot work for complex dataset. We believe

that our intuition can be extended to many other classes of preprocessing techniques and argue that they would not work under strong adversarial settings.

VIII. Conclusion

With all preprocessing defenses developed to date against test-time adversarial attacks on deep learning models called into question by recently proposed strong white box attacks, we take a first step towards understanding these defenses analytically. For this study, we focused on pixel discretization techniques, and, through a multi-faceted analysis, showed that if the base model has poor adversarial robustness, pixel discretization by itself is unlikely to improve robustness on any but the simplest datasets. Our study may pave the way for a broader understanding of the robustness of preprocessing defenses in general and guide how to design future preprocessing defenses.

IX. Acknowledgments

This material is partially supported by Air Force Grant FA9550-18-1-0166, the National Science Foundation (NSF) Grants CCF-FMitF-1836978, SaTC-Frontiers-1804648 and CCF-1652140 and ARO grant number W911NF-17-1-0405. Any opinions, findings, conclusions, and recommendations expressed herein are those of the authors and do not necessarily reflect the views of the funding agencies. Yingyu Liang would also like to acknowledge that support for this research was provided in part by the Office of the Vice Chancellor for Research and Graduate Education at the University of Wisconsin-Madison with funding from the Wisconsin Alumni Research Foundation.

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[2] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A study of the effect of jpg compression on adversarial images," *arXiv preprint arXiv:1608.00853*, 2016.

[3] C. Guo, M. Rana, M. Cissé, and L. van der Maaten, "Countering adversarial images using input transformations," *arXiv preprint arXiv:1711.00117*, 2017.

[4] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 135–147.

[5] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Proceedings of the 2018 Network and Distributed Systems Security Symposium (NDSS)*, 2018.

[6] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow, "Thermometer encoding: One hot way to resist adversarial examples," in *Submissions to International Conference on Learning Representations*, 2018.

[7] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," *arXiv preprint arXiv:1711.01991*, 2017.

[8] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," *arXiv preprint arXiv:1710.10766*, 2017.

[9] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," in *International Conference on Learning Representations*, vol. 9, 2018.

[10] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *arXiv preprint arXiv:1802.00420*, 2018.

[11] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits," http://yann.lecun.com/exdb/mnist/.

[12] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 39–57.

[13] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[14] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *6th European Machine Learning and Data Mining Conference (ECML/PKDD)*, 2013.

[15] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[16] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[17] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, no. 0, pp. –, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608012000457

[18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[19] A. Kurakin, I. Goodfellow, and S. Bengio, "Nips 2017: Defense against adversarial attack," https://www.kaggle.com/c/nips-2017-defense-against-adversarial-attack, 2017.

[20] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." in *AAAI*, vol. 4, 2017, p. 12.

[21] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[22] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Mądry, "Adversarially robust generalization requires more data," *arXiv preprint arXiv:1804.11285*, 2018.

[23] S. Boucheron, G. Lugosi, and P. Massart, *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.

[24] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *International Conference on Computer Aided Verification*. Springer, 2017, pp. 97–117.

[25] J. Z. Kolter and E. Wong, "Provable defenses against adversarial examples via the convex outer adversarial polytope," *arXiv preprint arXiv:1711.00851*, 2017.

[26] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," *arXiv preprint arXiv:1801.09344*, 2018.

[27] R. Perdisci, D. Dagon, W. Lee, P. Fogla, and M. Sharif, "Misleading worm signature generators using deliberate noise injection," in *Security and Privacy, 2006 IEEE Symposium on*. IEEE, 2006, pp. 15–pp.

[28] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing." in *USENIX Security Symposium*, 2014, pp. 17–32.

[29] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1322–1333.

[30] X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton, "A methodology for formalizing model-inversion attacks," in *Computer Security Foundations Symposium (CSF), 2016 IEEE 29th*. IEEE, 2016, pp. 355–370.

[31] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 3–18.

[32] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis." in *USENIX Security Symposium*, 2016, pp. 601–618.

[33] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016, pp. 372–387.

[34] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[35] S. M. Moosavi Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, no. EPFL-CONF-218057, 2016.