ROBUST ADVERSARIAL ℓ_p ATTACK DETECTION USING BLOCK-SPARSE DECOMPOSITION

Emre Acarturk
Department of Electrical and Computer Systems Engineering,
Rensselaer Polytechnic Institute,
Troy, NY 12180, USA
acarte@rpi.edu

Maruf A. Mridul & Thomas M. Waite
Department of Computer Science,
Rensselaer Polytechnic Institute,
Troy, NY 12180, USA
{mridum, waitet}@rpi.edu

ABSTRACT

Neural networks have consistently outperformed a variety of other models for high-dimensional classification tasks, particularly on image classification. However, with carefully-chosen, visually-imperceptible perturbations to their inputs, adversaries can almost always force the network to misclassify the image. One class of such attacks is the ℓ_p -norm bounded attacks, and while many methods have been developed to minimize the impact of such attacks on classification accuracy, fewer approaches have sought to detect and report potentially adversarial examples in a principled way. We take this defensive perspective in developing a robust attack detection and classification algorithms that utilize block-sparse signal decomposition to detect and report potentially adversarial inputs to a network, thereby providing additional user safety against attackers.

From Emre: Two things:

1. I have changed the tables to be in line with booktabs style. The old versions remain in comment blocks so it should be easy to revert.
2. I find the spacing around % weird. For now, I've consistently put a nonbreaking space before them. We can also just remove all spacing, it's up to you guys.

1 INTRODUCTION

Deep neural networks have been widely used in various applications, such as image classification, speech recognition, and natural language processing. However, it has been shown that these models are vulnerable to adversarial attacks, which are carefully crafted inputs designed to deceive the model into producing incorrect outputs(Biggio et al. (2013), Szegedy et al. (2014)). Adversarial attacks can be imperceptible to humans, making them a serious concern for security-sensitive applications.

Several defense methods have been proposed to mitigate the effects of adversarial attacks. However, these defenses have often been shown to be ineffective against new types of attacks. As a result, there is a continuous cycle of developing new attacks(Athalye et al. (2018), Carlini & Wagner (2017a), Uesato et al. (2018), Athalye & Carlini (2018)) and new defenses(Madry et al. (2017), Samangouei et al. (2018), Zhang et al. (2019), Papernot et al. (2016), Kurakin et al. (2016), Miyato et al. (2018), Zheng et al. (2016)), which has led to a cat-and-mouse game between attackers and defenders.

1.1 OUR CONTRIBUTIONS

Our work aims to investigate the utility of block-spare reconstruction of images into clean and perturbed components as a means to create a robust detection algorithm for ℓ_p norm bounded adversarial attacks. For this, we utilize a framework developed by Thaker et al. (2022) in which the authors claim that signals that have undergone ℓ_p norm bounded attacks can be reverse engineered to recover

the clean and perturbed components of the signal. Details for this framework are outlined in Section 3. Starting from this model, however, we contribute the following:

First, we aim to determine whether block-spare reconstruction can be used to accurately identify the presence and type of attack given a set of potentially perturbed signals. To achieve this we present attack detection and identification algorithms that leverage the block-spare reconstructed image and a neural network classifier identify potentially attacked images as well as classify the type among ℓ_p attack ($\ell_1, \ell_2,$ or ℓ_∞) for MNIST images.

Second, we aim to create a more robust detection algorithm, regardless of attack type, that minimizes the number of false positive attack detections. We show that this robust algorithm also finds the majority of attacks that are indeed successful on our network and can therefore be use to identify and remove potential adversaries from a corrupted data set, leading to significantly higher accuracy on the remaining data set.

Next, we present a survey of the related work in the field and summarize theoretical model on which our detection algorithms are based, then we define our algorithms and present the experimental results on the MNIST dataset.

2 Related Work

Structured representations have been successfully utilized in various applications such as image classification(Yang et al. (2009b), Mairal et al. (2008)), action recognition(Yang et al. (2009a), Castrodad & Sapiro (2012)), and speech recognition(Gemmeke et al. (2011), Sainath et al. (2011)) through sparse signal representation. This approach assumes that data from a particular class can be represented by a low-dimensional subspace based on training samples of that class. To classify new data, the correct sparse representation of the signal on the column space of a specific dictionary needs to be recovered. Despite the success of this approach, it fails to consider the challenges posed by adversarially corrupted inputs. The reference study of this project aims to investigate these challenges and their impact on the classification of structured data. (Wright et al. (2010), Mairal et al. (2008)) discuss on more examples of structured representations for data classification.

Recently, denoising-based defense strategies that utilize structured data representations have been proposed in the adversarial learning community. Notable examples of such approaches include the work of (Samangouei et al. (2018)) and (Moosavi-Dezfooli et al. (2018)), which have been extensively surveyed in (Niu et al. (2020)). However, these methods are limited in that they do not perform attack classification. In contrast, the approach in our reference paper jointly recovers the signal and attack, making it unique and novel from a theoretical standpoint. While the approach is not aimed at simply improving defense performance, it can still be compared to other defenses that aim to handle a union of perturbation families simultaneously, such as those developed in (Tramer & Boneh (2019), Maini et al. (2020), Croce & Hein (2019)).

Numerous literature exists on the detection of adversarial attacks. The problem aims to identify whether an example is an adversarial or clean example. Detection methods can be categorized as either unsupervised or supervised, as demonstrated in studies such as (Metzen et al. (2017)) and (Grosse et al. (2017)). (Bulusu et al. (2020)) did an extensive survey on these methods.

3 Theoretical Background

3.1 Reference Paper Summary

In this project, we build on the work of Thaker et al. (2022) on reverse engineering adversarial attacks. Specifically, given a corrupted signal $\mathbf{x}' = \mathbf{x} + \delta$, where $\mathbf{x}$ is a 'clean' signal and δ is an ℓ_p-norm bounded attack, the goal is to determine the attack type (ℓ_1, ℓ_2, or ℓ_∞) as well as the original signal $\mathbf{x}$. This problem is challenging since there could be many pairs $(\mathbf{x}, \delta)$ that yield the same $\mathbf{x}'$, and the attack vector δ may not be sparse.

To address these challenges, the authors propose to leverage results from the sparse recovery literature, which show that one can perfectly recover a signal $\mathbf{x}$ from a corrupted version $\mathbf{x}' = \mathbf{x} + \delta_0$ when both $\mathbf{x}$ and δ_0 are sparse on a meaningful basis. Specifically, if x is sparse with respect to

some signal dictionary D_s, and δ_0 is also sufficiently sparse, then the solution (c^*, δ^*) to the convex problem

$$\min_c ||\mathbf{c}||_1 + ||\delta||_1 \;\; s.t. \;\; \mathbf{x}' = D_s c + \delta \tag{1}$$

is such that $c^* = c_s$ and $\delta^* = \delta_0$. In other words, one can perfectly recover the clean signal as $x = D_s c^* = D_s x_s$ and the corruption δ_0 by solving the convex problem in (1).

However, these classical results from sparse recovery are not directly applicable to the problem of reverse engineering adversarial attacks. One of the challenges is that an attack δ may not be sparse, and while results from sparse recovery can be extended to bounded ℓ_2 errors(Candes et al. (2006)), they only guarantee stable recovery, instead of exact recovery, of sparse vectors close to the solution.

To overcome these challenges, the authors propose a block-sparse recovery problem, where both the signal and the attack are assumed to lie in a union of subspaces that includes one subspace per class and one subspace per attack type. They derive geometric conditions on the subspaces under which any attacked signal can be decomposed as the sum of a clean signal plus an attack. By determining the subspaces that contain the signal and the attack, they can also classify the signal and determine the attack type. Their framework is proposed under three main assumptions :

- The signal $\mathbf{x}'$ to be classified is the sum of a clean signal x plus an $\ell_p - norm$ bounded adversarial attack δ, i.e., additive attacks.
- The clean signal $\mathbf{x}$ is block sparse with respect to a dictionary of signals $\mathbf{D}_s$, i.e. $\mathbf{x} = \mathbf{D}_s \mathbf{c}_s$, where $\mathbf{D}_s$ can be decomposed into multiple blocks, each one corresponding to one class, and $\mathbf{c}_s$ is block sparse, i.e. $\mathbf{c}_s$ is only supported on a sparse number of blocks, but not necessarily sparse within those blocks.
- The $\ell_p - norm$ bounded adversarial attack also admits a block-sparse representation in the column space of a dictionary $\mathbf{D}_a$, which contains blocks corresponding to different ℓ_p bounded attacks.

The proposed approach is evaluated on digit and face classification tasks, and the experimental results demonstrate its effectiveness in recovering the clean signal and determining the attack type.

3.2 Mathematical Formalism

For this paper, we use the following setting and key optimization problems

Assume there is a clean signal $\mathbf{x}$ and an ℓ_p-bounded adversarial attack $\boldsymbol{\delta}$ that generate the observed signal $\mathbf{x}'$ exactly via addition, that is,

$$\mathbf{x}' = \mathbf{x} + \boldsymbol{\delta} \, .$$

Next, they assume that both $\mathbf{x}$ and $\boldsymbol{\delta}$ are *block-sparse*. In order to define this, they start by assuming that these signals are generated via

$$\mathbf{x} = \mathbf{D}_s \mathbf{c}_s \quad \text{and} \quad \boldsymbol{\delta} = \mathbf{D}_a \mathbf{c}_a \, ,$$

where $\mathbf{D}_{s,a}$ are dictionaries that are partitioned into blocks such that $\mathbf{D}_s$ has one block per output class, and $\mathbf{D}_a$ has one block per attack class. The optimization problem they propose is

$$\min_{\mathbf{c}_s, \mathbf{c}_a} \|\mathbf{c}_s\|_{1/2} + \|\mathbf{c}_a\|_{1/2} \quad \text{s.t.} \quad \mathbf{x}' = \mathbf{D}_s \mathbf{c}_s + \mathbf{D}_a \mathbf{c}_a \, , \tag{2}$$

where $\|\cdot\|_{1/2}$ is an ℓ_1/ℓ_2 norm hybrid that is designed to promote block sparsity on the coefficients $\mathbf{c}_{s,a}$, defined as having non-zero entries in only sparse number of classes while not being necessarily sparse within a class block. They also give a slightly different notation to make the notion of block sparsity more clear:

$$\mathbf{x} = \sum_{i=1}^{r} \mathbf{D}_s[i] \mathbf{c}_s[i] \quad \text{and} \quad \boldsymbol{\delta} = \sum_{i=1}^{r} \sum_{j=1}^{a} \mathbf{D}_a[i][j] \mathbf{c}_a[i][j] \, ,$$

where i corresponds to the signal class (of which there are r) and j corresponds to the attack class (of which there are a). Note that attacks are classified via both the attack type and signal class. This is because the authors construct the attack dictionary from the best adversarial attacks on the

classifier at the training samples. In this notation, the optimization problem in 2 can be stated only in terms of ℓ_2 norm as

$$\min_{\mathbf{c}_s,\mathbf{c}_a} \sum_{i=1}^{r} \|\mathbf{c}_s[i]\|_2 + \sum_{i=1}^{r}\sum_{j=1}^{a} \|\mathbf{c}_a[i][j]\|_2 \quad \text{s.t.} \quad \mathbf{x}' = \mathbf{D}_s\mathbf{c}_s + \mathbf{D}_a\mathbf{c}_a \,. \tag{3}$$

They also convert this constrained optimization problem to a regularized noisy version of it, given by

$$\min_{\mathbf{c}_s,\mathbf{c}_a} \|\mathbf{x}' - \mathbf{D}_s\mathbf{c}_s - \mathbf{D}_a\mathbf{c}_a\|_2^2 + \lambda_s \sum_{i=1}^{r} \|\mathbf{c}_s[i]\|_2 + \lambda_a \sum_{i=1}^{r}\sum_{j=1}^{a} \|\mathbf{c}_a[i][j]\|_2 \,. \tag{4}$$

This is the optimization problem we solve to generate the reconstructed image. For justification of its correctness and the algorithm used to solve it, we refer the reader to Thaker et al. (2022). Once solved, however, we are given the optimal $\mathbf{c}_s^*, \quad \mathbf{c}_a^*$, and we reconstruct the denoised image as follows:

$$\hat{i} = \arg\min_i ||\mathbf{x}' - \mathbf{D}_s[i]\mathbf{c}_s^*[i] - \mathbf{D}_a\mathbf{c}_a^*||_2 \tag{5}$$

$$\hat{\mathbf{x}} = \mathbf{D}_s[\hat{i}]\mathbf{c}_s^*[\hat{i}] \tag{6}$$

The result of this reconstruction process for an image under all three different attack methods is shown in Figure 1. We will return to the this formulation when determining attack classes, but the reconstructed image $\hat{\mathbf{x}}$ will allow us create a naive attack detection algorithm in the next section.

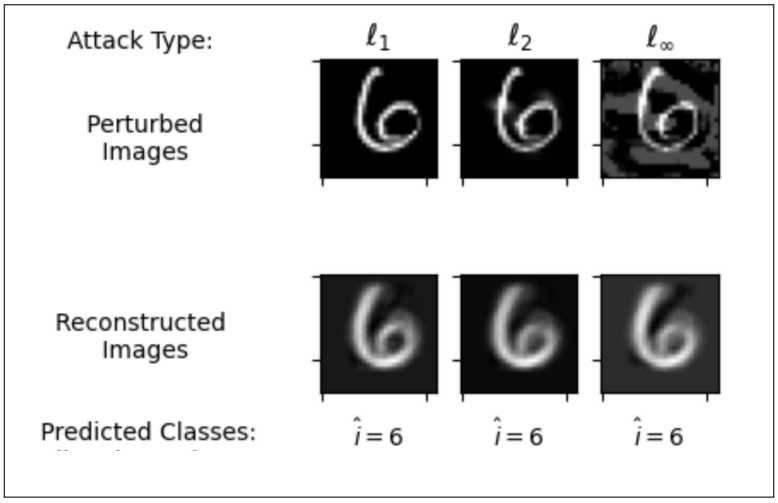

Figure 1: Sample Reconstructed images: $\hat{\mathbf{x}} = \mathbf{D}_s[\hat{i}]\mathbf{c}_s[\hat{i}]$ for each attack type.

4 Proposed Detection and Classification Framework

With this ability to reconstruct images into their signal and attack components, we propose the following approaches for attack detection and identification.

4.1 Attack Detection and Identification Algorithm

Given a set of potentially attacked data to be classified $\mathcal{X}_A$, for each image $\mathbf{x}_A$, we first compute its reconstruction $\hat{\mathbf{x}}_A$ according to equation (6). Then use our network classifier to predict both the

class of the potentially corrupted image $f(\mathbf{x}_A)$ and the reconstructed image $f(\hat{\mathbf{x}}_A)$. If they do not match, we save the image and its reconstruction to return to the user and we record a potential attack.

In addition simply detecting the presence of an attacked image, we seek to identify the type of attack carried out. Because we have a dictionary for the types of adversarial attacks, we can use it to determine the attack class using the following optimization problem:

$$\hat{j} = \arg\min_j ||\mathbf{x}' - \mathbf{D}_s\mathbf{c}_s^* - \mathbf{D}_a[\hat{i}][j]\mathbf{c}_a^*[\hat{i}][j]||_2 \tag{7}$$

This finds the attack index $\hat{j} \in \{0, 1, 2\}$ corresponding to the attack types $\ell_p \in \{\ell_1, \ell_2, \ell_\infty\}$ that makes minimizes the difference between the potentially attacked image and our reconstructed image with signal and attack noise components, corresponding to the most likely attack type. The specifics of the algorithm are outlined in Algorithm 1.

Algorithm 1 Our Attack Detection and Identification Algorithm

Inputs:
Corrupted Test Set: $\mathcal{X}_A$
NN Classifier: $f(\mathbf{x})$
Signal and Adversary Dictionaries: $\mathbf{D}_s,\ \mathbf{D_a}$

Initialize:
Attacked Image Set: $\mathcal{A} \leftarrow \{\}$

for all $\mathbf{x}_A \in \mathcal{X}_A$ **do**
Solve (4) to get $\mathbf{c}_s^*,\ \mathbf{c}_a^*$ with ASHA given $\mathbf{D}_a$, $\mathbf{D}_s$, and $\mathbf{x}_A$ ▷ See Thaker et al. (2022)
Compute $\hat{i}_A,\ \hat{\mathbf{x}}_A,$ and $\hat{j}_A$ according to Eqns. (5), (6), and (7)

/* If the predictions don't match, record attack */
if $f(\mathbf{x}_A) \neq f(\hat{\mathbf{x}}_A)$ **then**
$\mathcal{A} \leftarrow \mathcal{A} \bigcup \{\Big(\mathbf{x}_A,\ \hat{\mathbf{x}}_A, \hat{i}_A, \hat{j}_A\Big)\}$
end if
end for
return $\mathcal{A}$

4.2 Robust Attack Detection

The approach above is likely to produce many false positives, given that it must assume that the perturbed image underwent an attack to potentially identify the attack type via the reconstruction. In particular, if a given sample $\mathbf{x}'$ is not an attacked image, then the optimization and reconstruction process in Equations (5, 6) can introduce some additive attack noise from the dictionary of attacks. This effectively amounts to unintentionally adding some adversarial noise to the originally clean image, leading to misclassification by our network and therefore a false positive.

To instead only detect the presence of an attack rather than additionally identify its type and minimize the number of false positives, we reconstruct the image from its sparsity representation assuming it is not attacked. That is, we solve the following unconstrained optimization problem:

$$\mathbf{c}_s^* = \min_{\mathbf{c}_s} \|\mathbf{x}' - \mathbf{D}_s\mathbf{c}_s\|_2^2 + \lambda_s \sum_{i=1}^{r} \|\mathbf{c}_s[i]\|_2 \,. \tag{8}$$

Then we reconstruct the image as follows:

$$\hat{i} = \arg\min_i ||\mathbf{x}' - \mathbf{D}_s[i]\mathbf{c}_s^*[i]||_2 \tag{9}$$

$$\hat{\mathbf{x}} = \mathbf{D}_s[\hat{i}]\mathbf{c}_s^*[\hat{i}] \tag{10}$$

The algorithm is then almost identical to Algorithm 1, but we only solve the for $\mathbf{c}^*{}_s$, and we no longer require the adversarial dictionary. We also keep a set of the clean images (those for which we don't predict attack) to later show the improvement in accuracy on this clean set. The specifics are presented in Algorithm 2.

Algorithm 2 Our Robust Attack Detection Algorithm

Inputs:
Corrupted Test Set: $\mathcal{X}_A$
NN Classifier: $f(\mathbf{x})$
Signal Dictionary: $\mathbf{D}_s$

Initialize:
Attacked Image Set: $\mathcal{A} \leftarrow \{\}$
Cleaned Image Set: $\mathcal{X}_C \leftarrow \mathcal{X}_A$

for all $\mathbf{x}_A \in \mathcal{X}_A$ **do**
Solve (8) to get $\mathbf{c}_s^*$ with ASHA given, $\mathbf{D}_s$, and $\mathbf{x}_A$
Compute $\hat{i}_A,$ and $\hat{\mathbf{x}}_A$ according to Eqns. (9) adn (10)

/* If the predictions don't match, record attack and remove from clean image set*/
if $f(\mathbf{x}_A) \neq f(\hat{\mathbf{x}}_A)$ **then**
$\mathcal{A} \leftarrow \mathcal{A} \bigcup \{\Big(\mathbf{x}_A,\ \hat{\mathbf{x}}_A, \hat{i}_A\Big)\}$
$\mathcal{X}_C \leftarrow \mathcal{X}_C - \{\mathbf{x}_A\}$
end if
end for
return $\mathcal{A},\ \mathcal{X}_C$

5 Experiments

Our goal is to leverage the reconstructed, denoised images to robustly detect ℓ_p attacks against a neural network classifier $f(\mathbf{x})$. We present experiments on the MNIST dataset.

5.1 Experimental Context

Classifier and Attack Generator: For the MNIST dataset, we use a Convolutional Neural Network Classifier with architecture outlined in Table 3. It was trained for 5 epochs with a learning rate of 0.01, momentum of 0.5, and batch size of 128 images. The attack images were generated using projected gradient descent methods and implemented using the Advertorch library. The specific parameters for each type are included in Table 4.

Block Sparse Dictionary Construction: The signal and adversarial dictionaries ($\mathbf{D}_s$ and $\mathbf{D}_a$, respectively) were such that the columns for each class type are the flattened and normalized MNIST images for 200 sampled images from that class. In particular, to keep as small of set as possible, we implemented $\mathbf{D}_s$ as a $(10 \times 784 \times 200)$ tensor where $||\mathbf{D}_s[i,:,k]||_2 = 1\ \ \forall i,k$. Similarly, $\mathbf{D_a}$ is a $(3 \times 10 \times 784 \times 200)$ tensor where the first axis represents each of the three attack types. Sample entries form this dictionary are presented in Figure 4

Optimization Solver: To solve the optimization problem in Equation (4) that provides the coefficients to reconstruct the clean images, we used the cvxpy package with an SCS solver and 50 maximum iterations.

Metrics for Attack Detection and Identification: To evaluate the efficacy of our detector, we randomly perturbed a subset of the test data with a equal distribution of attack types across classes. In particular, we started with 1000 images drawn form the MNIST test set, then selected 30 images at random from each class to be perturbed. Finally, we assigned 10 of those to be perturbed by ℓ_1 attacks, 10 to be perturbed by ℓ_2 attacks, and 10 to be perturbed by ℓ_∞ attacks.

Given this perturbed set $:= \mathcal{X}_A$, we then run Algorithm (1) and use the returned set $\mathcal{A}$ to determine the following:

- **Overall Attacks Detected**: The percentage of the total real attacks that were detected by our algorithm and are found in the set $\mathcal{A}$.
- **True positives**: The number of real attacks detected in $\mathcal{A}$.

- **False positives**: The number of attacks detected in $\mathcal{A}$ that were not real attacks.
- **% ℓ_p Detected**: For each attack type, the percentage of the real ℓ_p type attacks detected. This shows which attack types are easier for our detector to detect.
- **Overall Attacks Correctly Identified**: Of the real attacks we detected correctly, the percent for which we also correctly identified the attack type.
- **% ℓ_p Identified**: Of the real attacks we detected correctly, the percent of each attack type we correctly identified. This show which attack types are easier for our detector to identify.

Metrics for Robust Attack Detection: Because our robust algorithm only seeks to compute the number of attacks detected and does not compute a predicted class type, we can instead measure its efficacy across attack types aside from basic ℓ_p bounded PGD attacks. In particular, we extend the attacks to include Carlini-Wagner L2 attack from Carlini & Wagner (2017b), a Momentum Iterative Method (MIM) from Dong et al. (2018), and a Decoupled Direction and Norm (DDN) ℓ_2 attack by Rony et al. (2019). Moreover, this method can only be used on non-PGD attack. To demonstrate this, we use a Spatial Transform (ST) attack that is not ℓ_p bounded by Xiao et al. (2018). This attack seeks to make images that still look like the image class to the user, but utilizes spatial transforms to create adversaries. All of these attacks were implemented from the Advertorch library using their default parameters.

Finally, to evaluate the efficacy of this more general tool, we determine the following additional accuracy metrics:

- **Accuracy on True Positives (TP)**: Of the detected real attacks, the percent correctly classified by the network.
- **Accuracy on False Negatives (FN)**: Of the real attacks our detector missed, the percent correctly classified by the network.
- **Accuracy on $\mathcal{X}_C$**: Of the images not returned by our detector, the percent correctly classified by the network.
- **Raw Accuracy** : The percent of images correctly identified by the network on the entire perturbed set.

The metrics above allow two key analyses of our detector. The accuracy of true positives vs false negatives will allow us to see if the detector finds the stronger attacks present it the perturbed set. That is, the attacks that are most likely to be misclassified. The accuracy on the raw images vs the cleaned images (the set with our detected attacks removed) demonstrates how removing the detected attacks from an incoming stream of images will impact the classifiers accuracy on the remaining images.

5.2 Attack Detection and Identification Results and Discussion

Table 1 shows the results of three different trials of varying attack parameter strength.

Discussion of results: In general, our algorithm was able to detect approximately 40 % of the attacks, with an slightly more false positives than true positives on average. In addition, the detector is on average more effective at detecting and identifying ℓ_∞ type attacks as it typically found a majority of such attack as compared to about 30 % of the other two types.

For identification, the algorithm was able to correctly identify the attack type on about half of the attacks it correctly detected. This was slightly higher with very high attack epsilons, but the majority of these correct identifications came from ℓ_∞ attacks. Intuitively, this makes sense as the ℓ_∞ attacks look more dissimilar from the other types at high epsilon values. See Figure 4 for a visualization of the different attack types.

Qualitatively, Figure 2 demonstrates how an example of true positive, false positive, and false negative images with their reconstructions and predicted classes.

Table 1: Detection and identification results for 300 attacked images of 1000 test images from the MNIST data set with varying attack parameters. 10 images were selected at random from each class to be attacked by each attack type.

Attack Parameters	Overall Detected	True Positives	False Positives	ℓ_1 **Detected**	ℓ_2 **Detected**	ℓ_∞ **Detected**	Overall Identified	ℓ_1 **Identified**	ℓ_2 **Identified**	ℓ_∞ **Identified**
$\epsilon_{\ell_1} = 5, \epsilon_{\ell_2} = 1, \epsilon_{\ell_\infty} = .15$	41 %	122	172	32 %	24 %	66 %	43 %	59 %	29 %	39 %
$\epsilon_{\ell_1} = 10, \epsilon_{\ell_2} = 2, \epsilon_{\ell_\infty} = .3$	43 %	130	117	21 %	30 %	79 %	45 %	52 %	23 %	52 %
$\epsilon_{\ell_1} = 20, \epsilon_{\ell_2} = 4, \epsilon_{\ell_\infty} = .6$	37 %	111	155	22 %	31 %	58 %	63 %	55 %	26 %	86 %

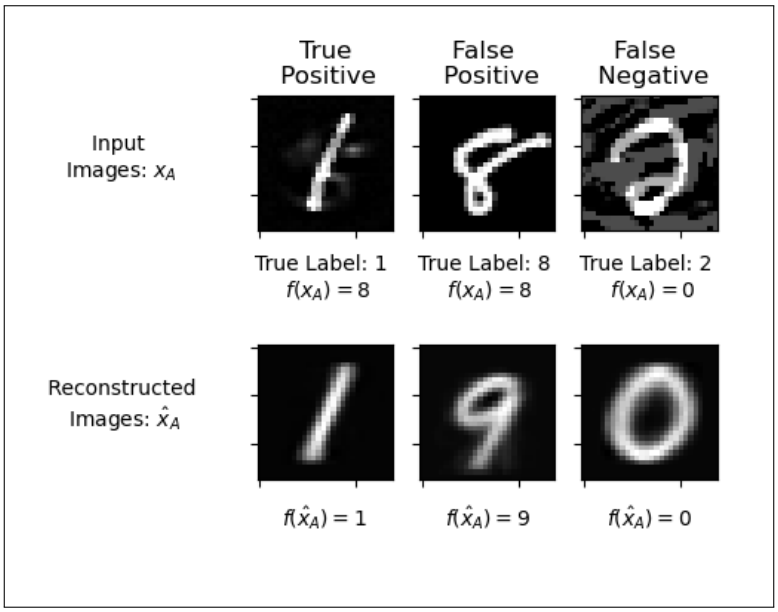

Figure 2: Sample true positive, false positive and false negative images with their reconstructions and predicted classes on

5.3 Robust Attack Detection Results:

Table 2 shows the results of our robust detection algorithm with a variety of attack types, strengths, and sizes.

Overall Results:
Overall, the robust detection algorithm resulted in a detection rate of approximately 30% for the basic PGD attacks analyzed previously. Unlike the detection and identification algorithm, however, the robust detection found many fewer false positives. In fact, the number of false positives does not seem to be correlated with the number of attacked images and is instead a feature of the subset of test images used for a given trial. Some qualitative examples of how the detect and identification algorithm was producing false positives can been seen in Figure 3.

Accuracy on TP vs FN, and $\mathcal{X}_C$ vs $\mathcal{X}_A$:
Not only did the robust detector find adversaries with fewer false positives, it also found the images that are most likely to result in misclassification. Because the adversaries were capped in their number of iterations to be generated, not all were true adversaries of the network. When this was

the case, our algorithm found the adversaries that were likely to be misclassified. This translates to a much higher accuracy on the set of clean images returned by the algorithm as shown in the last 2 columns of the table.

Detection Robustness:
Attacks that were not basic PGD ℓ_p bound attacks were more detectable by our algorithm. Specifically, CW, DDN, and Spatial Transform attacks all produced fully adversarial examples, and our detection rates for these attacks were significantly higher, leading to a much higher accuracy on the cleaned images set $\mathcal{X}_C$. In all of these cases, the size of the detected set of images $\mathcal{X}_A - \mathcal{X}_C$ was either smaller that the number of actual attacks or close to the same size. For example, the DDN attack produced the largest attack set of size 343, but this is still close the actual number of attacks of 300.

Table 2: Robust detection results using various attack types on 1000 MNIST test images. The attacks with multiple types indicated are assigned uniformly across classes and classes. The single type attacks are applied at random uniformly across classes.

Attack Parameters	Total Attacks	Overall Detected	True Positives (TP)	False Positives (FN)	Acc. on TP	Acc. on FN	Acc. on $\mathcal{X}_C$	Raw Acc.
$\epsilon_{\ell_1} = 5, \epsilon_{\ell_2} = 1, \epsilon_{\ell_\infty} = .15$	300	29 %	86	63	10 %	93 %	98 %	89 %
$\epsilon_{\ell_1} = 10, \epsilon_{\ell_2} = 2, \epsilon_{\ell_\infty} = .3$	300	35 %	105	59	7 %	72 %	93 %	84 %
$\epsilon_{\ell_1} = 20, \epsilon_{\ell_2} = 4, \epsilon_{\ell_\infty} = .6$	300	23 %	68	48	13 %	59 %	89 %	84 %
$\epsilon_{\ell_1} = 10, \epsilon_{\ell_2} = 2, \epsilon_{\ell_\infty} = .3$	600	34 %	205	32	5 %	77 %	88 %	72 %
$\epsilon_{\ell_1} = 10, \epsilon_{\ell_2} = 2, \epsilon_{\ell_\infty} = .3$	150	35 %	52	58	8 %	76 %	96 %	92 %
CW	300	91 %	274	37	0 %	0 %	96 %	69 %
MIM	300	60 %	180	51	0 %	20 %	87 %	72 %
DDN	300	93 %	278	65	0 %	0 %	95 %	69 %
Spatial Transform	300	86 %	259	66	0 %	0 %	93 %	69 %

6 FUTURE WORK

While our detection algorithm shows promise as a general purpose tool for adversarial attack detection on images, there are many further directions to explore. Currently, a major issue is the consistent presence of false positives. There are two clear next steps to attempt to reduce the number of false positives.

First, a significant hyperparameter in this problem is the size of the signal dictionary used to reconstruct the images. Initially, 200 was an appropriate choice given that we additionally had to store $3 \times 10 \times 200$ adversarial signals in addition. Given the optimization problem for the robust detection algorithm does not require the adversarial dictionary, it might be possible to use a larger signal dictionary, leading to better reconstruction and likely fewer still false positives. Or similarly, there has been work determine core sets of training data or finding the most important training samples for efficient and robust network training. See Paul et al. (2021) and Mirzasoleiman et al. (2020). It would be interesting to explore if such core sets could be used to construct the signal dictionary to maintain robust reconstructions, even with relatively small dictionary sizes.

A second approach could be to train the network with reconstructed training images. Augmenting the training data with reconstructions would help ensure correct classifications of unaltered images, further reducing false positives.

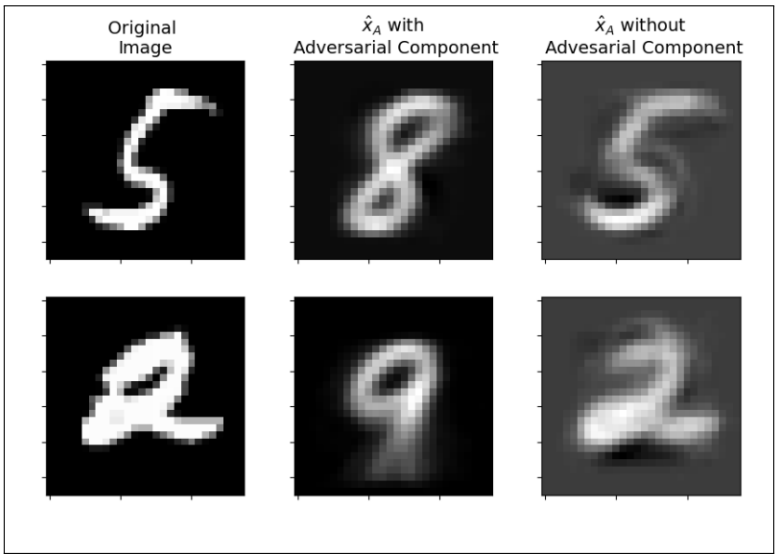

Figure 3: Examples of the robust algorithms reconstruction reducing the number of false positives. The two images are from classes 2 and 5, respectively, and the robust reconstruction without the adversarial component recovers and image that is classified correctly, whereas reconstruction with noise leads to a misclassification of 8 and 9, respectively.

Some additional considerations are that all experiments were conducted with untargeted attacks, but additional testing should be done to see if targeted attacks have any influence on the detector's performance.

7 Conclusion

In this paper, we studied how block sparse reconstruction can be utilized to detect adversarial attacks in potentially corrupted datasets. We provided two algorithms to this end and evaluate their performance on the MNIST dataset. We first presented a detection and identification algorithm that uses decomposition into signal and adversarial components to classify the attack type as well as denoise the image. This approach is limited to only the types of attacks seen in the adversarial dictionary, and induces many false positives on unattacked images. We then presented a robust detection algorithm that decomposes the signal without assuming attack and reconstructs it from a signal dictionary to then be classified by the network. This approach greatly reduced the number of false positives and correctly detected a majority of adversaries, particularly when those adversaries are most effective. We believe there are many interesting directions to further consider reconstruction techniques as a means of detecting out of distribution or adversarial examples.

REFERENCES

Anish Athalye and Nicholas Carlini. On the robustness of the cvpr 2018 white-box adversarial example defenses. *arXiv preprint arXiv:1804.03286*, 2018.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pp. 274–283. PMLR, 2018.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*, pp. 387–402. Springer, 2013.

Saikiran Bulusu, Bhavya Kailkhura, Bo Li, Pramod K Varshney, and Dawn Song. Anomalous example detection in deep learning: A survey. *IEEE Access*, 8:132330–132347, 2020.

Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8):1207–1223, 2006.

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pp. 3–14, 2017a.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks, 2017b.

Alexey Castrodad and Guillermo Sapiro. Sparse modeling of human actions from motion imagery. *International journal of computer vision*, 100:1–15, 2012.

Francesco Croce and Matthias Hein. Provable robustness against all adversarial ℓ_p-perturbations for $p \geq 1$. *arXiv preprint arXiv:1905.11213*, 2019.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum, 2018.

Jort F Gemmeke, Tuomas Virtanen, and Antti Hurmalainen. Exemplar-based sparse representations for noise robust automatic speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2067–2080, 2011.

Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Pratyush Maini, Eric Wong, and Zico Kolter. Adversarial robustness against the union of multiple perturbation models. In *International Conference on Machine Learning*, pp. 6640–6650. PMLR, 2020.

Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Discriminative learned dictionaries for local image analysis. In *2008 IEEE conference on computer vision and pattern recognition*, pp. 1–8. IEEE, 2008.

Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models, 2020.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

Seyed-Mohsen Moosavi-Dezfooli, Ashish Shrivastava, and Oncel Tuzel. Divide, denoise, and defend against adversarial attacks. *arXiv preprint arXiv:1802.06806*, 2018.

Zhonghan Niu, Zhaoxi Chen, Linyi Li, Yubin Yang, Bo Li, and Jinfeng Yi. On the limitations of denoising strategies as adversarial defenses. *arXiv preprint arXiv:2012.09384*, 2020.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pp. 582–597. IEEE, 2016.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34:20596–20607, 2021.

Jérôme Rony, Luiz G. Hafemann, Luiz S. Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses, 2019.

Tara N Sainath, Bhuvana Ramabhadran, Michael Picheny, David Nahamoo, and Dimitri Kanevsky. Exemplar-based sparse representation features: From timit to lvcsr. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(8):2598–2613, 2011.

Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, IJ Goodfellow, and Rob Fergus. Intriguing properties of neural networks: Proceedings of the international conference on learning representations. 2014.

Darshan Thaker, Paris Giampouras, and René Vidal. Reverse engineering ℓ_p attacks: A block-sparse optimization approach with recovery guarantees. In *International Conference on Machine Learning*, pp. 21253–21271. PMLR, 2022.

Florian Tramer and Dan Boneh. Adversarial training and robustness for multiple perturbations. *Advances in neural information processing systems*, 32, 2019.

Jonathan Uesato, Brendan O'donoghue, Pushmeet Kohli, and Aaron Oord. Adversarial risk and the dangers of evaluating against weak attacks. In *International Conference on Machine Learning*, pp. 5025–5034. PMLR, 2018.

John Wright, Yi Ma, Julien Mairal, Guillermo Sapiro, Thomas S Huang, and Shuicheng Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044, 2010.

Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples, 2018.

Allen Y Yang, Roozbeh Jafari, S Shankar Sastry, and Ruzena Bajcsy. Distributed recognition of human actions using wearable motion sensor networks. *Journal of Ambient Intelligence and Smart Environments*, 1(2):103–115, 2009a.

Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *2009 IEEE Conference on computer vision and pattern recognition*, pp. 1794–1801. IEEE, 2009b.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pp. 7472–7482. PMLR, 2019.

Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pp. 4480–4488, 2016.

A APPENDIX

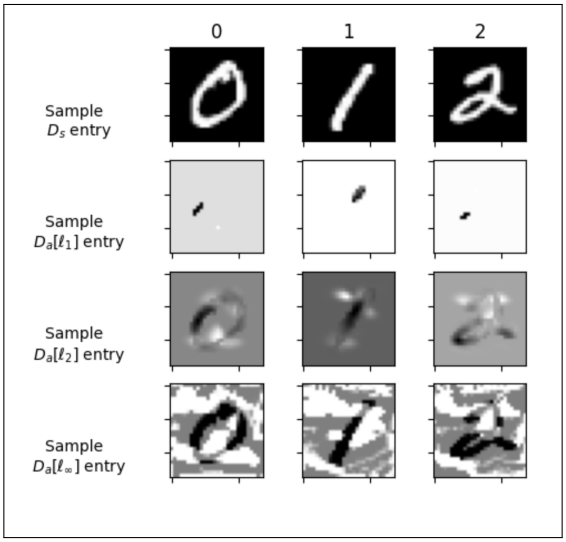

Figure 4: Sample Signal and Adversarial Perturbation Dictionary Entries. Note that images are scaled to be unit size, so the pixel intensity of the perturbed images does not represent the actual size of the generated perturbation. They are included here to demonstrate how the images are organized and the qualitative impact of each attack type.

Table 3: Network Architecture For MNIST Classifier

Layer Type	**Size**
2x(Conv2d+ReLu)	$3 \times 3 \times 32$, stride = 1
MaxPool2D	2×2, stride=1
2x(Conv2d+ReLu)	$3 \times 3 \times 64$, stride=1
MaxPool2D	2×2, stride=1
2x(Linear+ReLu)	200
Linear+ReLu	10

Table 4: Aversarial Attack Parameters

Attack Type	**Max Perturbation** ϵ	**Step Size** α	**Max Iterationr Num** N
ℓ_1	10	0.8	100
ℓ_2	2	0.1	200
ℓ_∞	0.3	0.01	100