FAST AND ROBUST DEEP NEURAL NETWORKS DESIGN

A Thesis presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

---

by

ZHIQUN ZHAO

Dr. Zhihai(Henry) HE, Thesis Supervisor

DEC 2020

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled:

FAST AND ROBUST DEEP NEURAL NETWORKS DESIGN

presented by Zhiqun Zhao,
a candidate for the degree of Doctor of Philosophy and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Zhihai(Henry) He

Dr. Yunxin Zhao

Dr. Kannappan Palaniappan

Dr. Ming Xin

ACKNOWLEDGMENTS

I would like to sincerely thank my advisor Prof. Zhihai He, the chair of this thesis. It was such a privilege to have him as my advisor. Throughout my doctoral work, he encouraged me to develop research skills. He continuously simulated my analytical thinking and greatly assisted me with scientific writing. I would like to thank Prof. Hengyou Wang, who provide so many helpful advice and encourage me to move on during my PhD life. I would also like to thank all the other committee members of my dissertation, Prof. Yunxin Zhao, Prof. Kannappan Palaniappan and Prof. Ming Xin for their precious advice and comments.

I would also like to thank my family. I could not make it so far without my parents' support and encouragement. I would thank all my colleagues as well for their help of work and life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

In the past few years, we have witnessed a rapid development of deep neural networks in computer vision, from basic image classification tasks to some more advanced applications *e.g.* object detection and semantic segmentation. Inspire of its great success, there exists two challenges of deep neural networks real-world applications: its computational cost and vulnerability. Thus we are aimed to deal with these two problems in this thesis.

To speed up deep networks, we propose a $\mathcal{L}_1$-Norm based low-rank approximation method to reduce float operations based on the alternating direction method (ADM) in Chapter 2. Our experimental results on public datasets, including CIFAR-10 and ImageNet, demonstrate that this new decomposition scheme outperforms the recently developed $\mathcal{L}_2$-norm based nonlinear decomposition method.

To defend against adversarial examples, we develop a novel pre-processing alogrithm based on image restoration to remove adversarial attack noise in Chapter 3. We detect high-sensitivity which have significant contributions to the image classification performance. Then we partition the image pixels into the two groups: high-sensitivity and low-sensitivity keypoints. For the low-sensitivity pixels, we use the existing total variation (TV) norm-based image smoothing. For the high-sensitivity pixels, we develop a structure-preserving low-rank image completion methods. Based on matrix analysis and optimization, we have derived an iterative solution for this optimization problem. This high-sensitivity points detection helps us to improve the defense against white-box attack *BPDA*.

However, in our keypoints defense we only remove and recover a few part of pixels,

which indicates there are still many perturbation over the whole image. In Chapter 4, we propose a novel image completion algorithm structure-preserving progressive low-rank image completion (*SPLIC*) based on smoothed rank function (SRF) in which we can reconstruct a image with over 50% removed pixels. In *SPLIC*, we randomly remove over 50% pixels on the image and then do matrix completion by low-rank approximation to remain the global structure of the image. Differ from other low-rank methods, we replace nuclear norm by smoothed rank function (SRF) for its closer rank function approximation. We introduce total variance (TV) regularization to improve image reconstruction, and then combine total variance (TV) norm de-noising to further remove the perturbation over the whole image. Then we train the network on the *SPLIC* images. The experimental results show our *SPLIC* outperforms other pre-processing methods in image reconstruction, gray-box and black-box scenario.

Chapter 1

Introduction

1.1 Background

First of all, a brief background knowledge of image representations including Deep Convolutional Neural Networks (DCNNs) and low-rank Approximation will be introduced as follow.

1.1.1 Deep Convolutional Neural Networks

Recently, deep learning, as a branch of machine learning based on artificial neural networks with multiple functional layers including non-linear transformations, has become the baseline approach in various computer vision tasks. Inspired by biological neural networks, researchers created artificial neural networks (ANN) in 1940s and the theory was completely developed well in 1970s. Due to recent development of digital electronics hardware and the huge data emergence, deep learning has achieved a great

success in both academia and industry. Convolutional neural network (CNN), as the most commonly used network model, has played a very important roles in almost all computer vision tasks.

Convolutional Layers

The convolutional layer is the most important functional component of a convolutional neural network which has most of the computational complexity. It contains a set of filters with learnable parameters. Each of these filters is small spatially, but extends through the full depth of the input tensor. A typical first layer convolutional filter has size of $7 \times 7 \times 3$ (i.e. 7 pixels width and height, 3 for RGB color channels). In the forward operation, we slide each filter across the width and height of the input and compute dot products of the entries of the filter and the input at all position. Then we can obtain a 2D feature map that gives the responses of the sliding filter at every spatial position. In the backward operation, the filter parameters will be updated by the gradients of the loss function. Intuitively, filters can be activated when they see some visual features. For examples, the first layer filters might recognize some local features like an edge of some orientation or a blotch of some color, and filters on higher layers might recognize some global patterns like entire honeycomb or wheel.

Fully-connected Layers

Unlike convolutional layer, fully connected layers connect every neuron on the current layer to every neuron in the next layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). Usually, the neurons' activation will be computed with a matrix multiplication followed by a bias offset.

1.1.2 Low-rank Approximation

In the low rank approximation, a given matrix M can be approximated by a low-rank matrix $\hat{M}$, *i.e.* $\|M - \hat{M}\| \approx 0$. This area is also known as principal component analysis. A high dimensional data can be represented as a much lower dimensional data. But the point of low-rank approximation is not necessarily just for performing dimension reduction. In the ideal case where the entries of matrix M are not affected by noise, corruption, missing values etc. But in reality, the observed matrix typically will have much higher rank. Low-rank approximation is thus a way to recover the original (the "ideal" matrix before it was corrupted by noise) low-rank matrix, *i.e.* find the matrix that is most consistent with the current matrix and is low-rank so that it can be used as an approximation to the ideal matrix. Low-rank approximation is a minimization problem in mathematics. It's cost function measures the fit between a given matrix and an approximated matrix, which is subject to a constraint that the approximated matrix has reduced rank. The low-rank approximation technique is usually used in mathematical modeling and data compression.

1.2 Deep Neural Networks Acceleration

During the past several years, deep neural networks, especially deep convolutional neural networks (DCNNs), have achieved remarkable success in various computer vision tasks including classification [3, 4], detection [5, 6] and segmentation [7, 8]. The continuously improving performance often builds upon increasingly huge volume of labeled data, deeper networks, and massive numbers of network parameters, often in the range of tens or even hundreds of millions, which result in prohibitively high

computational complexity [4, 9]. This creates significant challenges in deploying these high-performance networks in real-world devices, platforms, and applications. For example, portable devices such as smart phones and tablets have limited computational power. Cloud service platforms have access to very powerful GPUs. But, each cloud service needs to handle a large number of concurrent requests, which results in very limited computational resources for each deep neural network task. In this case, the available computing resource for each deep neural network task becomes very limited. Therefore, accelerating the deep neural networks has become an urgent research task.

In a deep convolutional neural networks (DCNNs), convolutional layers contribute to the most computational complexity [10, 11]. Therefore, most of the existing methods for accelerating the DCNNs have been focusing on optimizing the convolution layers of the network [12, 13].

A set of approaches have been developed in the literature, including connection pruning and low-rank approximation [14, 12]. Normal connection pruning produce non-structured network connections, resulting in irregular memory access that often degrades the performance of network speed optimization, especially in practical implementations on parallel computing devices, such as GPUs [13, 14].

Low-rank approximation has emerged as a promising approach for speeding up DCNNs. Typical high-performance DCNNs have large numbers of network filters and channels, whose parameters are to be learned during the training process. The low-rank approximation approach aims to explore the redundancy or correlation existing between different network channels and filters, approximate and speed up the network using low-rank decomposition [15, 16]. All these approximations are based on $\mathcal{L}_2$-norm. Whereas, many recent low-rank approximation and image reconstruction works

[17, 18, 19] show $\mathcal{L}_2$-norm approximation is very sensitive to outliers and will cause large approximation error.

To deal with the outliers problem, we propose a new $\mathcal{L}_1$-norm based method to reduce the rank of filters for DCNNs acceleration and develop a mathematical solution for this optimization problem. We also develop a *iterative two-step procedure* to fine-tune the whole network after each decomposition and further improve the performance. We demonstrate that the linear $\mathcal{L}_1$-norm decomposition is more efficient than the non-linear $\mathcal{L}_2$-norm decomposition in accelerating deep neural networks in our experimental results.

1.3 Deep Neural Networks De-noising

Recently, researchers have realized that deep neural networks are very sensitive to adversarial attacks [20]. A little changes of input image can easily fool the deep neural network image classifier. Because this small error at the input layer can be gradually increased along the network inference layers, finally exceed the decision boundary at the last layer, and result in the false decision [20, 21]. This vulnerability of deep neural networks has become a critical threat in many real-world applications of deep neural networks, such as face recognition, security monitoring, and autonomous driving [22].

There exists three different adversarial attack modes: *white-box*, *gray-box* and *black-box*. In white-box attacks, the attacker knows all details of the classifier network including the network architecture and model parameters, and the details of defense strategy as well. While gray-box attack can only access the classifier. In the case of

black-box attack [23], neither the classifier and the defense strategy are visible, but a substitute classifier can be trained to mimic the real classifier behavior and then generates adversarial perturbations by applying attack algorithms on the substitute classifier.

The adversarial attacks can be considered as a special type of image noise. Recently, a number of image de-noising methods [24, 25, 26] have been developed to remove adversarial attack noise and recover the image. These noise removal-based approaches are attractive because they do not introduce any changes to the network. More importantly, unlike other defense methods based on deep neural network design or adversarial training, they cannot be easily attacked by existing white-box attack methods. This is because most of existing white-box attacks are based on gradient back propagation and the gradient cannot be easily propagated through the highly complicated image de-noising algorithms.

Adversarial attacks generate the perturbation on every point of the image. But impacts of those perturbations at each point are not equal. We propose a hyper-thesis that even in the case of *FGSM* [20], in which magnitudes of all perturbations are the same, the impacts of different points can not be consistent.

Based on this hyper-thesis, we a new approach to defending image against adversarial noise in Chapter 3. Firstly, we rank all points in the input image by their contributions to the classification result. Based on this ranking result, we can obtain a mask which differentiates the high-sensitivity keypoints and low-sensitivity keypoints. To address them discriminately, low-sensitivity keypoints are processed by TV norm minimization de-noising [27], while the high-sensitivity keypoints are removed and a new image will be recovered by our reweighted low-rank matrix approach. Our exten-

sive experimental resutls on benchmark datasets demonstrate that our approach can achieve highly effective defense and outperforms existing noise removal-based defense methods with powerful black-box, gray-box, and white-box attacks.

1.4 Deep Neural Networks Robustness

As mentioned in 1.3, many image pre-processing studies [24, 25, 26] are naturally considered to be the strategy for perturbations removal. However, they are challenged by the *BPDA* attack [28], since these defenses essentially provide obfuscated gradients and can be broken by approximated gradients. Then Yang *et al.* [29] points out that human recognize images by the global structure, while image processing by computers, especially deep neural networks, are more likely affected by the local structure. They develop *ME-Net* to randomly discard over 50% pixels and then reconstruct the image by matrix completion. The matrix completion in *ME-Net* is formulated as a rank minimization problem. In this way, computers can learn the images global structure by the reconstructed images.

The completion algorithm in *ME-Net* is based on nuclear norm, which is the tightest convex relaxation of the rank minimization problem. Recently, many studies replace nuclear norm by *smoothed rank function (SRF). SRF* methods have the advantage that smooth term lie much closer than nuclear norm to rank function, which results in the better reconstruction. However, the original *SRF* performance often degrades due to the yielded noise and the reconstructed image is lack of structure smoothness. Another weakness of *ME-Net* is that there still remains some adversarial perturbation because of the preserved pixels.

To deal with this problem. We propose a novel *TV-regularization* constrain into the *SRF* and develop the mathematical solution for this optimization problem. With the help of the TV-regularization the image reconstruction performance is further improved. We also exploit the TV-norm to smooth the whole image, which not only benefits the image reconstruction but also increase the resistance of adversarial perturbations. Our experiment demonstrates our *TVSRF* reconstruction outperforms the *ME-Net* on almost all situation and we achieve a much better defense than other data pre-processing methods.

1.5 Summary

We recognize deep learning real-world implementation still faces two challenges: the speed and the security. Low-rank approximation, as a effective and robust data dimension reduction method, can be explored to solve deep learning application problems. In this thesis, we study the deep learning acceleration and adversarial robustness problems and provide solutions by exploiting low-rank approximation. In the first part of our work, we introduce our layer decomposition algorithm which is based on $\mathcal{L}_1$-norm and show its speedup result. In the second part of our work, we present a defending strategy against adversarial attacks based on high-sensitivity points detection and demonstrate the significant improvement. This defending strategy only involves image de-noising. In the third of our work, we propose a novel image pre-processing method by introducing TV smoothness to train a robust deep convolutional neural network classifier, which achieve a great success against many kinds of adversarial attacks.

The rest of this thesis is organized as follows. In Chapter 2, we introduces the deep neural networks acceleration algorithm. In Chapter 3, we presents our de-noising defense based on high-sensitivity keypoints selection. In Chapter 4, we show our new smoothed rank function method with TV smoothness and discuss the future work.

Chapter 2

$\mathcal{L}_1$-Norm Low-Rank Linear Approximation for Accelerating Deep Neural Networks

2.1 Motivation

Recently, to accelerate the test-phase computation of DCNNs, Zhang *et al.* [12] developed a network speed optimization method which decomposes the convolution layer into two convolution layers using $\mathcal{L}_2$-norm based low-rank approximation. They also considered the joint approximation of the convolution and nonlinear activation (e.g. ReLU) layers so that the method can be applied to deeper networks. However, recent studies on low-rank matrix approximation and image recovery have demonstrated that $\mathcal{L}_2$-norm approximation is sensitive to outliers, which may result in a large approximation error. To address this issue, many efficient low-rank approximation methods developed for image recovery often resort to $\mathcal{L}_1$-norm [17, 18, 19] analysis. Motivated

by its success in image recovery, in this work, we propose to develop an $\mathcal{L}_1$-norm based low-rank decomposition method for approximating the convolution layers so as to speed up the DCNN.

Specifically, in this paper, we propose a linear *Low-Rank Approximation of Responses (LRAR)* algorithm based on $\mathcal{L}_1$-norm for accelerating the test-phase computation of deep convolutional neural networks to improve the accuracy of accelerated networks in [12]. As illustrated in Fig. 2.1. We decompose one pre-trained convolutional layer's filters into two groups: the first one is the basic filters and the second one is linear combination filters. Once the decomposition of the trained network is finished, the decomposed network can be used directly in the test-phase. During our $\mathcal{L}_1$-norm based low-rank decomposition, we aim to minimize the approximation error between the original output Y without decomposition and the output $\hat{Y}$ after decomposition. This new $\mathcal{L}_1$-norm based low-rank decomposition problem can be solved by an augmented Lagrange method. After the decomposition of the convolution layer, we fine-tune the network to improve its accuracy. Our experimental result demonstrates that this new low-rank decomposition can significantly reduce the computational complexity of the convolution layers and speed up the whole network. We also demonstrate that the proposed $\mathcal{L}_1$-norm decomposition is more efficient than the $\mathcal{L}_2$-norm based decomposition recently developed in the literature [12].

2.2 Related Work

Convolution layers are the most time-consuming component in deep convolutional neural networks [30]. Currently, the most widely-used method to implement convolu-

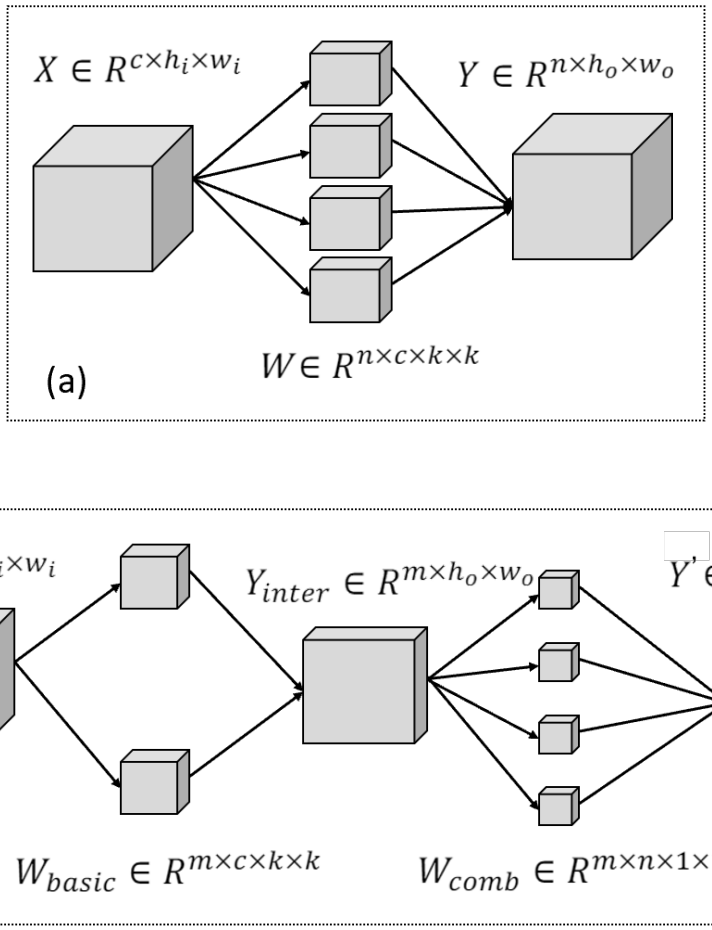


Figure 2.1: (a) Conventional convolution layer before decomposition; (b) convolutional layer after decomposition. The weight shape of the original layer is $n \times c \times k \times k$. It is decomposed as two layers: W_{basic} is the weight of the basic filter and its shape is $m \times c \times k \times k$, and W_{comb} is the weight of the linear combination filter and its shape is $n \times m \times 1 \times 1$, where $m < n$. Y_{inter} is the intermediate output of the basic convolutional layer and $\hat{Y}$ is the approximated output. Then the complexity can be reduced from $n \times (h_o w_o) \times (k^2 c)$ to $m \times (h_o w_o) \times (k^2 c) + n \times (h_o w_o) \times m$.

tion is based on matrix multiplication, which is often accelerated by the Basic Linear Algebra Subprograms (BLAS) libraries, such as Intel MKL and OpenBLAS [31]. To further speed up the convolution layers, connection pruning and low-rank approximation are two commonly used approaches. Connection pruning exploits matrix sparsity to save computations [30]. Low-rank approximation reduces the rank of the matrix to save computations [12]. In the following, we provide a detailed review of these two approaches and other related methods.

2.2.1 DCNNs Acceleration by Connection Pruning

Connection pruning was firstly proposed by Han *et al.* to reduce the DCNN parameters [30]. Deep compression [10] introduced a DCNN model compression algorithm with three stages: pruning, trained quantization, and Huffman coding. These methods focused on fully connected layers for model compression. Many recent methods shift the focus to convolution layers by pruning their connections to achieve computation reduction. Wen *et al.* [11] exploited group-lasso regularization during DCNN training to obtain filters with structured sparsity. Lebedev *et al.* developed a similar idea in [32]. ThiNet [9] introduced a filter-level pruning method and argued that the pruning criteria should be the output of its next layer instead of the current layer. Yu *et al.* [14] developed a three-step method for pruning: first, it scored every neuron in the final response layer for its importance; the importance scores were then propagated through the whole network; finally, the method pruned the neurons based on these importance scores. Zhao *et al.* [33] leverage the batch normalization layer to prune the channels with small coefficient. Generative adversarial learning (GAL) is proposed to solve the pruning optimization problem in [34].

2.2.2 DCNNs Acceleration by Low-rank Approximation

Low-rank approximation has been extensively used in matrix recovery, matrix completion, and low-rank representation [35, 36, 37]. The traditional principal component analysis (PCA) [38, 39] evaluated the approximation error based on Frobenius norm, which was effective for Gaussian additive noise. [35, 40] demonstrated that $\mathcal{L}_1$-norm based low-rank decomposition, or called robust PCA, is robust to large and sparse noise or outliers. Recent studies on low-rank matrix theory [41, 42] mostly focus on $\mathcal{L}_1$-norm analysis. The main objective of these methods was to decompose the noisy matrix into a low-rank matrix and a sparse error matrix. Specifically, it can be formulated into the following minimization problem:

$$\begin{aligned} &\min_{X,E} \big\{ \|X\|_* + \lambda \|E\|_1 \big\}, \\ s.t. \quad &E = D - X, \end{aligned} \tag{2.1}$$

where $\|\cdot\|_*$ is the nuclear norm, the sum of singular values of the matrix.

Recently, researchers are exploring various applications of low-rank approximation in DCNN optimization [43] by decomposing one single high-complexity convolutional layer into several low-complexity convolutional layers. Jaderberg *et al.* [15] demonstrated that the redundancy between feature channels and filters can be exploited to speed up the convolutional neural networks. [15] designed two simple schemes and two optimization functions to reduce redundant channels and filters. Denton *et al.* [44] proposed two alternative methods: monochromatic approximation and bi-clustering approximation. Force regularization developed in [16] exploited the correlation among filters during the DCNNs training stage to ensure the low-rank properties of filters.

Zhang *et al.* [12] considered both convolution layers and nonlinear layers, such as the ReLU layers of DCNN, and solved the nonlinear low-rank approximation problem so as to optimize deep neural networks. They exploited the ideas of multi-layer asymmetric reconstruction, rank selection, higher-dimensional decomposition, and fine-tuning. During our experiments, we observed that this non-linear approximation often destroys the well-learned feature pattern of the network layer and causes degraded weights during fine-tuning. Inspired by [12] and motivated by the success of $\mathcal{L}_1$-norm based low-rank decomposition for image restoration, in this work, we propose a $\mathcal{L}_1$-norm low-rank decomposition scheme to approximate high complexity convolution layers and speed up the whole network in test-phase. Our experimental results demonstrate that this $\mathcal{L}_1$-norm based linear approximation significantly reduces the approximate errors, especially with subsequent fine-tuning.

2.2.3 Other Related Methods

Besides low-rank approximation and connection pruning, quantization is another way to achieve both network compression and acceleration. Hubara *et al.* [45] proposed Quantized Neural Networks (QNNs). [13, 46, 47] demonstrated that converting the neural networks into binary forms can obtain much smaller network models and improve the inference speed. Instead of pruning weights in convolutional layers, Figurnov *et al.* [48] only computed the important position of inputs, which was essentially equivalent to pruning inputs. [49] developed four guidelines for light-weight networks architecture design. MobileNets [50] developed a light-weight network for mobile devices. SqueezeNet [51] designed a smaller network and shuffleNet [52] reduced the computation cost based on pointwise group convoultion and channel shuffle. Shuf-

fleNet V2 [49] analyzed major contributing factors of DCNN computation, such as floating computation, memory access cost (MAC), and platform characteristics.

2.3 Method

In this section, we first analyze the computational complexity of convolution layers in DCNNs and then explain how the low-rank approximation can speed up a convolutional layer. Based on this analysis, we propose our $\mathcal{L}_1$-norm based low-rank approximation of convolutional layers.

2.3.1 Matrix Formulation of Convolution

In deep convolutional neural networks, the convolution operation can be formulated by the following matrix multiplication:

$$Y = WX + B, \tag{2.2}$$

where $Y \in \mathbb{R}^{n \times h_o \times w_o}$ is the output tensor, $W \in \mathbb{R}^{n \times k^2 \times c}$ is the weight tensor, $X \in \mathbb{R}^{(k^2 c) \times (h_o w_o)}$ is the input tensor, and $B = \{\vec{b_1}, \vec{b_2}, \cdots, \vec{b_n}\}^T$, $B \in \mathbb{R}^{n \times (h_o w_o)}, \vec{b_i} \in \mathbb{R}^{(h_o w_o) \times 1}$, is the bias tensor. Here, h_o and w_o are the height and width of the output, c is the number of input channels, n is the number of output channels (*i.e.* the number of filters), and k is the convolution kernel size. The bias B and weight W can be concatenated together into one matrix $W \in \mathbb{R}^{n \times (k^2 c+1)}$. In this way, the input $X \in \mathbb{R}^{(k^2 c+1) \times (h_o w_o)}$ is formed by concatenating original input and a vector of all ones

(*i.e.* $\vec{1} \in \mathbb{R}^{1\times(h_o w_o)}$). Then, the convolution can be rewritten as:

$$Y = WX, \tag{2.3}$$

where $Y \in \mathbb{R}^{n\times(h_o w_o)}$, $W \in \mathbb{R}^{n\times(k^2c+1)}$, and $X \in \mathbb{R}^{(k^2c+1)\times(h_o w_o)}$. Ignoring the addition operation on biases which has very small computatonal complexity, we can calculate number of floating operations (FLOPs) of a convolutional layer, by

$$\mathbf{C_{orig}} = n \times (h_o w_o) \times (k^2 c). \tag{2.4}$$

2.3.2 Low-rank Approximation of Convolutional Layers

The low-rank decomposition of the convolution layer builds upon the assumption that there exists correlation among output channels or the output space is a low-rank subspace, and they can be represented by a linear combination of fewer basic channels. Let $\vec{y_i} \in \mathbb{R}^n$ be the i-th column of Y. According to the analysis in [12], we have

$$\begin{aligned} \vec{y_i} &= M\vec{y_i} + \vec{b'}, \\ s.t. \quad & rank(M) < n, \end{aligned} \tag{2.5}$$

where M is an $n \times n$ low-rank matrix, and b' is the bias vector. Then, M can be written as $M = PQ^T$, $P \in \mathbb{R}^{n\times m}$ and $Q \in \mathbb{R}^{n\times m}$. Here m is the rank of M which is

less than n. According to the above analysis, (2.5) can be written as

$$\begin{aligned} \vec{y_i} &= PW'\vec{x_i} + \vec{b'}, \\ s.t. \quad & rank(M) < n, \end{aligned} \tag{2.6}$$

where $W' = Q^T W$ is the new matrix of basic filters. $P \in \mathbb{R}^{n \times m}$ is the weight matrix for linear combination of these basic filters to form the approximation output. In this way, the single convolutional layer is decomposed into two convolutional layers: one convolutional layer with the basic filters and one linear combination convolutional layer whose kernel size is 1×1. The computational complexity, measured by number of floating operations (FLOPs), of these two decomposed small convolutional layers is given by

$$\mathbf{C_{decomp}} = m \times (h_o w_o) \times (k^2 c) + n \times (h_o w_o) \times m. \tag{2.7}$$

The acceleration of DCNNs requires that

$$\mathbf{C_{decomp}} < \mathbf{C_{orig}} \tag{2.8}$$

This low-rank approximation reduces the convolution computation by decomposing one convolutional layer with high computational complexity into two convolutional layers with very low computational complexity.

2.3.3 $\mathcal{L}_1$-Norm Low-Rank Decomposition

In this section, we present our proposed $\mathcal{L}_1$-norm based low-rank decomposition for DCNNs speedup. Let $X = \{x_1, x_2, \cdots^{\rightarrow}, x_{h_o \times w_o}\}$ be the matrix of $h_o \times w_o$ input

vectors to the convolution layer. Based on the assumption that the output $Y = \{y_1, y_2, \cdots^{\rightarrow}, y_{h_o \times w_o}\}$ lies in a low-rank subspace, we can rewrite (2.5) as

$$Y = MY + B' \overset{a}{=} MY + \bar{Y} - M\bar{Y} = M(Y - \bar{Y}) + \bar{Y}, \tag{2.9}$$

The equality $\overset{a}{=}$ is obtained by set $B' = \{\vec{b'}, \vec{b'}, \cdots, \vec{b'}\} \in \mathbb{R}^{n \times (h_o w_o)}$ and $\vec{b'} = \bar{y} - M\bar{y}$, where $\bar{Y} = \{\bar{\vec{y}}, \bar{\vec{y}}, \cdots, \bar{\vec{y}}\} \in \mathbb{R}^{n \times (h_o w_o)}$ with $\bar{\vec{y}}$ as the mean vector of the response Y. From the above equation (2.9), it can be concluded that if the output space is one low-rank subspace, then we can write $Y - \bar{Y} = M(Y - \bar{Y})$. But in fact, the output space just located in an approximated low-rank subspace, so we try to find a low-rank subspace, so that the error of $\|Y - \bar{Y} - M(Y - \bar{Y})\|$ as small as possible.

Mathematically, to find an approximated low-rank subspace for the response, we can minimize the rank of matrix M, which is equivalent to minimizing its nuclear norm. In addition, to make our model robust to outlier noise, we wish its error matrix to be as sparse as possible, which can be evaluated by its $\mathcal{L}_1$ norm. In addition, when approximation is done in each layer independently, the error will be gradually increased from shallower layers to deeper layers and finally affect the output layer. We followed the asymmetric reconstruction method to alleviate this problem as [12] and denote the approximated output to the current layer as $\hat{Y}$. We can compute its non-approximate responses as $Y = WX$ for the training data. So we try to minimize the error of $\|Y - \bar{Y} - M(\hat{Y} - \bar{\hat{Y}})\|_1$ in stead of $\|Y - \bar{Y} - M(Y - \bar{Y})\|_1$. In this way, it can be reformulated as the following optimization problem:

$$\begin{aligned} &\min_{\{M,E\}} \big\{\|M\|_* + \lambda\|E\|_1\big\}, \\ s.t. \quad &E = Y - (M\hat{Y} + B''), \end{aligned} \tag{2.10}$$

where $\|\cdot\|_*$ is the nuclear norm of matrix and $\|\cdot\|_1$ is the $\mathcal{L}_1$ norm. Here, $B'' = \{\vec{b}'', \vec{b}'', \cdots, \vec{b}''\} \in \mathbb{R}^{n\times(h_o w_o)}$, and $\vec{b}'' = \bar{\vec{y}} - M\bar{\hat{\vec{y}}}$, where $\bar{\hat{\vec{y}}}$ is the mean vector of the approximated responses $\hat{Y} = \{\hat{y_1}, \hat{y_2}, \vec{\cdots}, \hat{y}_{h_o w_o}\}$. Substituting B'' into the constraint $E = Y - (M\hat{Y} + B'')$, we have

$$\begin{aligned} E &= Y - (M\hat{Y} + \bar{Y} - M\bar{\hat{Y}}) \\ &= Y - \bar{Y} - M(\hat{Y} - \bar{\hat{Y}}) \end{aligned} \tag{2.11}$$

where $\hat{Y} \in \mathbb{R}^{n\times(h_o w_o)}$ and $\bar{\hat{Y}} = \{\bar{\hat{\vec{y}}}, \bar{\hat{\vec{y}}}, \cdots, \bar{\hat{\vec{y}}}\} \in \mathbb{R}^{n\times(h_o w_o)}$. Hence, we can rewrite the optimization problem in (2.11) as:

$$\begin{aligned} &\min_{\{M,E\}} \big\{\|M\|_* + \lambda\|E\|_1\big\}, \\ s.t. \quad &E = Z - MH, \end{aligned} \tag{2.12}$$

where $Z = Y - \bar{Y}$ and $H = \hat{Y} - \bar{\hat{Y}}$.

2.4 Solving the Optimization Problem

In this section, based on the alternating direction method (ADM), we aim to develop a mathematical solution for the optimization problem in (2.12) to obtain $\mathcal{L}_1$-norm low-rank approximation of the convolution layer. To this end, we introduce the following augmented Lagrange function, which adds the constraint into the objective function

of problem (2.12):

$$\min_{\{M,E,L\}} \big\{ \|M\|_* + \lambda \|E\|_1 + < L, Z - MH - E > + \frac{\mu}{2} \|Z - MH - E\|_F^2 \big\}, \qquad (2.13)$$

where $Z = Y - \bar{Y}$ and $H = \hat{Y} - \bar{\hat{Y}}$. L is the matrix of Lagrange multipliers. $< \cdot, \cdot >$ is the inner product of two matrices. It should be noted that it is very difficult to obtain a close-form solution for the problem in (2.13). In this work, we propose an iterative solution method base on the ADM, which optimizes the sub-problem with one variable while fixing the rest variables. In this way, the complex problem of multiple variables can be divided into several reduced optimization problems with one single optimization variable which can be solved analytically. Finally, we demonstrate that this iterative method converges very fast for our convolution layer approximation problem. Note that, in (2.13), there are three optimization variables (M, E, L). In the following, we derive the mathematical solution to optimize each of these three variables. It should be noted that the optimization problem in (2.13) for low-rank decomposition of convolution layers is significantly different from those in image restoration [41, 42]. Therefore, we need to drive new mathematical solutions for this $\mathcal{L}_1$-based low-rank decomposition problem, which will be explained in detail in the following sections.

2.4.1 Optimizing the Low-Rank Matrix M

The first variable to be optimized is M, which is a low-rank matrix. In (2.13), if we fix variables (E, L), then the matrix M can be optimized as follows:

$$\begin{aligned}
&\arg\min_{M} \quad \big\{\|M\|_* + \lambda\|E\|_1 + < L, Z - MH - E > + \frac{\mu}{2}\|Z - MH - E\|_F^2\big\} \\
\stackrel{a}{=} &\arg\min_{M} \quad \big\{\|M\|_* + < L, Z - MH - E > + \frac{\mu}{2}\|Z - MH - E\|_F^2\big\} \\
\stackrel{b}{=} &\arg\min_{M} \quad \big\{\|M\|_* + < L, -MH > + \frac{\mu}{2} < Z - MH - E, Z - MH - E > \big\} \\
\stackrel{c}{=} &\arg\min_{M} \quad \big\{\|M\|_* + \frac{\mu}{2}\|MH - (Z - E + \frac{L}{\mu})\|_F^2\big\}.
\end{aligned} \tag{2.14}$$

The equality $\stackrel{a}{=}$ and $\stackrel{b}{=}$ of (2.14) can be obtained by reducing those items without M and expanding the squared item of the Frobenius norm. The equality $\stackrel{c}{=}$ is obtained by reconstructing the new squared item of the Frobenius norm. We recognize that this problem (2.14) is hard to be solved directly. Generally, the target of the minimization problem (2.14) is to find a low-rank matrix M because it contains the item $\|M\|_*$, otherwise it just finds a matrix M, which maybe have a higher rank. In our proposed low-rank approximation for DCNNs acceleration, the speed is related with the rank of M. Thus, to control the speed, the rank can be given at first. If the rank of the matrix M is given, then the first item will not affect the solution. Hence this optimization problem can be simplified as:

$$\begin{aligned}
&\arg\min_{M} \big\{\frac{\mu}{2}\|MH - (Z - E + \frac{L}{\mu})\|_F^2\big\} \\
= \quad &\arg\min_{M} \big\{\|MH - (Z - E + \frac{L}{\mu})\|_F^2\big\}.
\end{aligned} \tag{2.15}$$

This problem in (2.15) is well known as the Reduced Rank Regression, which belongs

to a broader category of procrustes problems [53, 54]. This problem can be solved with generalized singular value decomposition (GSVD) [12, 54]. The problem in (2.15) has the full rank solution $\hat{M} = WH^T(HH^T)^{-1}$, where $W = Z - E + \frac{L}{\mu}$. Let $\hat{M} = USV^T$ be its GSVD. The Reduced Rank solution is given by $M = U_m S_m V_m^T$, where U_m and V_m are the first m columns of U and V, and S_m are the largest m singular values. Thus, the low-rank matrix is given by:

$$M = U_m S_m V_m^T. \tag{2.16}$$

Now, the most important question is how to obtain the GSVD of M. To this end, we use the method developed in [55]. First, we have the following Lemma.

Lemma 1. *Let K and Q be metric matrices. Let A be an matrix of rank m. $R_K^T A R_Q = R_K^T U S V^T R_Q$ is the GSVD of A under metrics K and Q. R_K and R_Q are square root factors of K and Q, respectively. We have $U^T K U = I$ and $V^T Q V = I$. If the usual SVD of $R_K^T A R_Q$ is presented as $U^* S^* V^{*T}$, then the GSVD of A under metrics K and Q can be computed as $U = (R_K^T)^{-1} U^*$, $V = (R_Q^T)^{-1} V^*$ and $S = S^*$.*

Compared with the above Lemma 1, in our problem, K is an identity matrix I, $Q = H$, and $A = \hat{M} = WH^T(HH^T)^{-1}$, so if we have the SVD of $R_I^T \hat{M} R_H$ to be $U^* S^* V^{*T}$, then we can compute the GSVD of $\hat{M}$ by setting $U = U^*$, $V = (R_H^T)^{-1} V^*$ and $S = S^*$. In this way, the matrix M with rank m can be computed by (2.16).

2.4.2 Optimizing the Error Matrix E

The second variable to be optimized is E, which is the error matrix. In (2.13), if we fix variables (M, L), then the matrix E can be optimized as follows:

$$\begin{aligned}
&\arg\min_{E} \quad \{\|M\|_* + \lambda\|E\|_1 + < L, Z - MH - E > + \frac{\mu}{2}\|Z - MH - E\|_F^2\} \\
&\stackrel{a}{=} \arg\min_{E} \quad \{\lambda\|E\|_1 + < L, Z - MH - E > + \frac{\mu}{2}\|Z - MH - E\|_F^2\} \\
&\stackrel{b}{=} \arg\min_{E} \quad \{\lambda\|E\|_1 + < L, -E > + \frac{\mu}{2} < Z - MH - E, Z - MH - E > \} \\
&\stackrel{c}{=} \arg\min_{E} \quad \{\lambda\|E\|_1 + \frac{\mu}{2}\|E - (Z - MH + \frac{L}{\mu})\|_F^2\}.
\end{aligned} \tag{2.17}$$

The equality $\stackrel{a}{=}$ and $\stackrel{b}{=}$ of (2.17) are obtained by reducing the items without E and expanding the squared item of the Frobenius norm. The equality $\stackrel{c}{=}$ is obtained by reconstructing the new squared item of the Frobenius norm. Finally, the optimization problem of (2.17) is transformed to a well known problem, which can be solved by the iterative shrinkage-thresholding operator [56]. Let $Q = Z - MH + \frac{L}{\mu}$, matrix E can be updated using the following formula:

$$E = \mathcal{S}_{\frac{\lambda}{\mu}}(Q), \tag{2.18}$$

where $\mathcal{S}_{\frac{\lambda}{\mu}}(Q)$ has the solution $sgn(q_{ij}) \cdot max\{q_{ij} - \frac{\lambda}{\mu}, 0\}$ [56].

2.4.3 Updating of the Lagrange Multiply Matrix L

L is the Lagrange multiplier matrix. If L is the optimization variable and the rest variables are considered as constants, the first-order derivation of (2.13) equals to

zero, then L can be updated as follows:

$$L = L + \mu(Z - MH - E). \tag{2.19}$$

2.5 Algorithm Summary

In this section, we summarize the proposed algorithm for solving the $\mathcal{L}_1$-norm low-rank approximation of convolution layers in Algorithm 1. The relative error of the observed matrices is used as the stopping criteria:

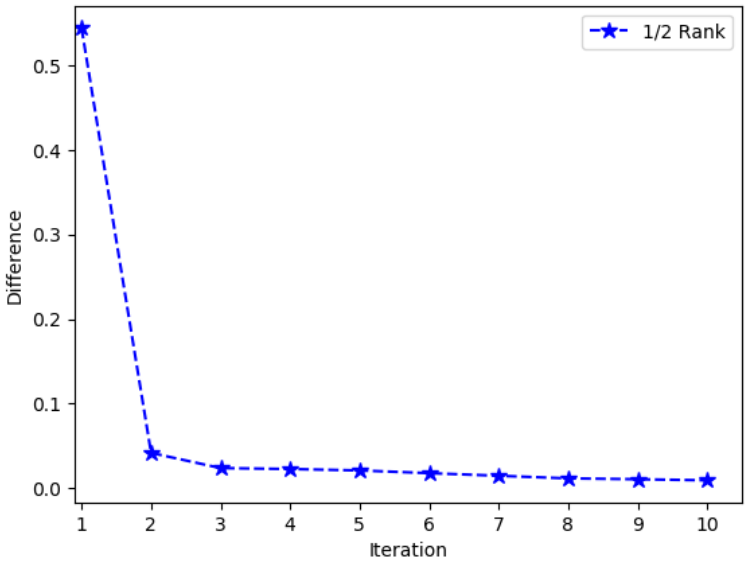
$$E_r = \frac{\|Z - MH - E\|_F}{\|Z\|_F} < \varepsilon, \tag{2.20}$$

where ε is set to be a small positive number. In our experiments, inspired by [35] and [57], to obtain better results, the parameter μ is adjusted by $\delta = 1.1$ in each iteration.

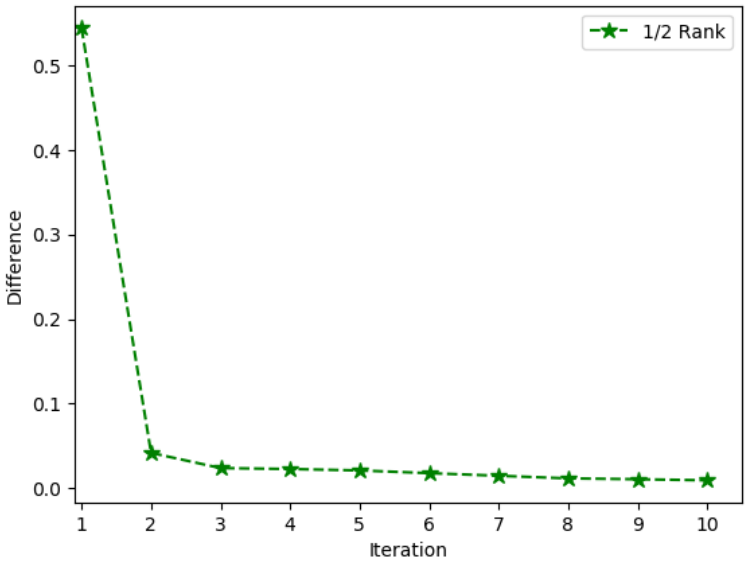
Algorithm 1 Linear $\mathcal{L}_1$-Norm Low-Rank Approximation

Require: $Y, \hat{Y} \in R^{m \times n}$, λ
Ensure: Set $Z = Y - \bar{Y}$, $H = \hat{Y} - \bar{\hat{Y}}$, $M = O_{m \times n}$,
$E = O_{m \times n}$, $L = \frac{H}{\|H\|_F}$, $\delta = 1.1$, $\mu = 10^{-3}$,
$\mu_{max} = 10^{10}$.
1: **while** $\frac{\|Z - MH - E\|_F}{\|Z\|_F} > \varepsilon$ **do**
2: Compute M based on (2.16).
3: Compute E based on (2.18).
4: Compute L based on (2.19).
5: Compute μ as $\mu = min(\delta\mu, \mu_{max})$.
6: **end while**
7: **Output** M

We test the convergence of our method on a simple hand-written digit data. The approximation is applied on the first layer of pre-trained LeNet [58] model. The

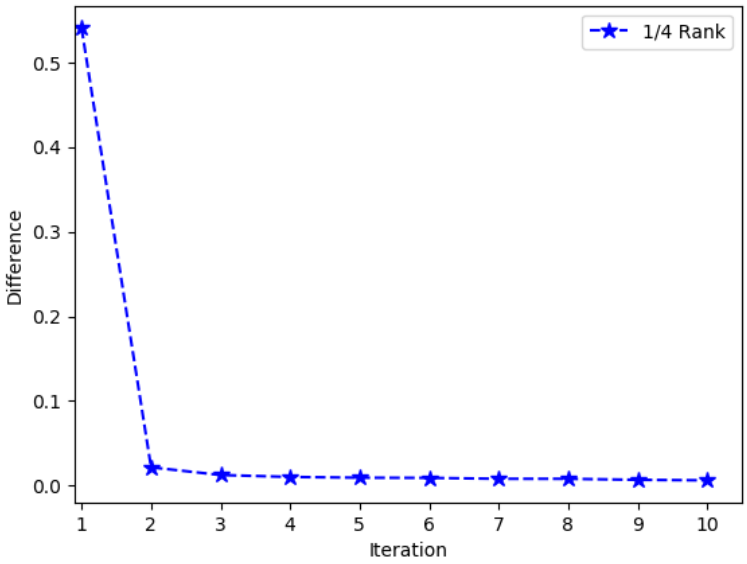


(a) M 1/2 rank

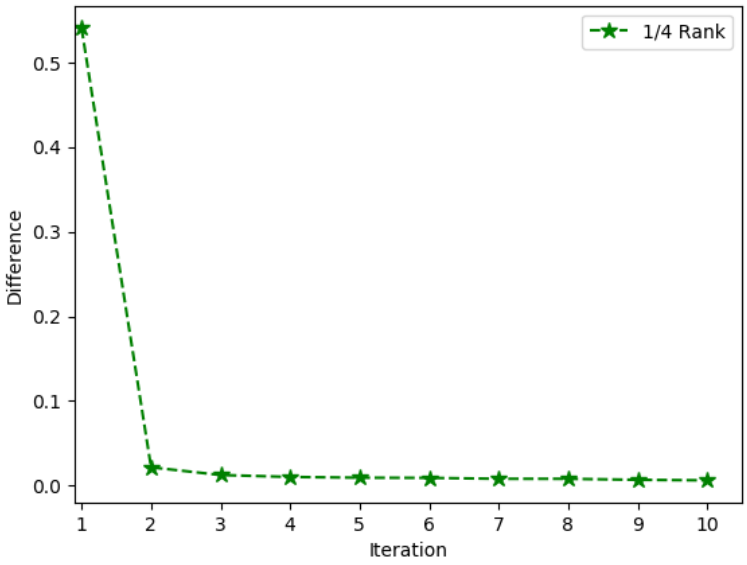


(b) E 1/2 rank

Figure 2.2: Convergence analysis at 1/2 rank: the top two show the difference between M_t and M_{t-1}, and the bottom two show the difference between E_t and E_{t-1}



(a) M 1/4 rank



(b) E 1/4 rank

Figure 2.3: Convergence analysis at 1/4 rank: the top two show the difference between M_t and M_{t-1}, and the bottom two show the difference between E_t and E_{t-1}

convergence of the ADM algorithm for the smoothed objective function has been generally proven in [59] and [60]. Up to the present, it is still difficult to generally ensure the convergence of our proposed method. Since the objective function of (2.12) is not smooth, it would be not easy to prove the convergence in theory. Fortunately, there actually exist some guarantees for ensuring the convergence of Algorithm 1. According to the theoretical results in [61], some conditions are sufficient (but may not necessary) for Algorithm 1 to converge: one of the conditions is that the gap produced in each iteration step is monotonically deceasing. As shown in Figure 2.2 and 2.3, the differences of M_t with M_{t-1} and E_t with E_{t-1} (t is the iterative step) are both monotonically deceasing. Furthermore, the nuclear norm and $\mathcal{L}_1$-norm are both convex functions. Thus, our proposed objective function is an convex optimization problem. And our proposed method is global convergence.

The approximation is totally separated from the network test-phase computation. Once low-rank approximation is done, the DCNNs test-phase computation complexity is fixed. Although our $\mathcal{L}_1$-norm method may spend more time than the $\mathcal{L}_2$-norm method because of the iterative solution during approximation, our error is smaller than the $\mathcal{L}_2$-norm method. The high complexity of our method trades high performance of the approximated network. Therefore, our optimization can achieve higher classification accuracy with the same complexity of networks test-phase computation.

2.6 Experimental Results

In this section, we evaluate the performance of our LRAR algorithm on image classification tasks and compare its performance with existing $\mathcal{L}_1$-based non-linear approx-

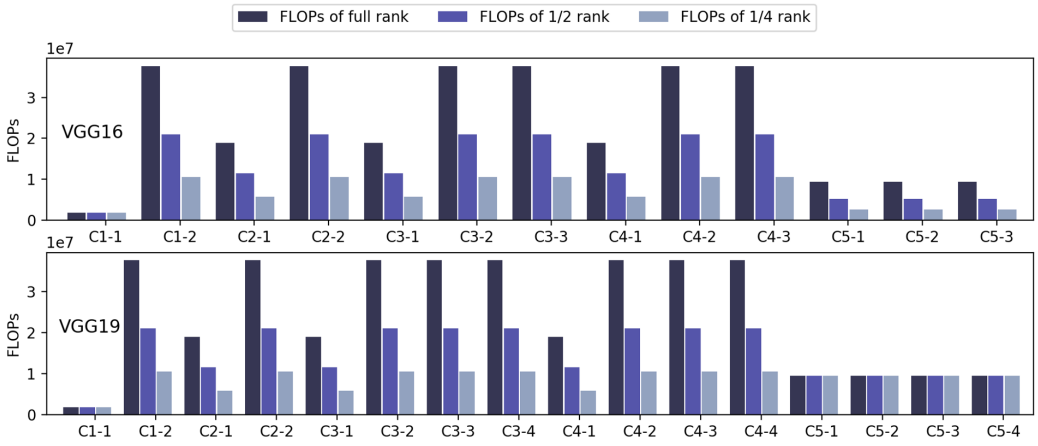


Figure 2.4: Computational complexity of different network layers measured in FLOPs of VGGNet at full rank, and with $\mathcal{L}_1$-norm low-rank decomposition of 1/2 rank and 1/4 rank.

imation [12] which is referred to as the NL2A algorithm. We compare the accuracy of the approximated networks obtained by different methods. These networks have the same computational complexity specified by the decomposition ratio, such as 1/2 or 1/4. To examine the approximation error of the convolution layer, we will compute the Frobenius norm of the residual between the approximated output and the original one. Two benchmark datasets are selected for performance evaluations: the CIFAR-10 [62] consisting of 60,000 32×32 images and ILSVRC-2012 [63] consist-

ing of 1,781,167 large scale images. We evaluate the approximation performance on *CIFAR-10* with a large network *VGG* [43]. But, on the large *ILSVRC-2012* dataset, we choose a relatively small network network *AlexNet* [3]. Otherwise, if the a large network, such as the VGG, is being used, the amount of computing time is tremendous. We use the above two different configurations to evaluate the performance of the our proposed method with comparison against existing state-of-the-art method. We aim to demonstrate that the linear $\mathcal{L}_1$-norm low-rank approximation is more efficient than the non-linear $\mathcal{L}_2$-norm low-rank approximation.

During low-rank approximation, we randomly take 100,000 output points from the training data in each convolution layer to assemble the output matrix $Y \in \mathbb{R}^{n \times 100{,}000}$. After low-rank decomposition of the convolution layers, we also perform fine-tuning of the network and compare the performance. We observe that the fine-tuning process is sensitive to the learning rate: a large learning rate may cause the training loss not to converge and a small learning rate will result in slow convergence of the training loss. In this work, we set the learning rate to be 1×10^{-5}. The network is retrained for 30 epochs after low-rank approximation and the learning rate is configured to decay by a factor of 0.1 after 20 epochs. We set the batch-size as 128.

All algorithms are implemented using PyTorch [64] running on a desktop computer with an Intel core i7-7800X 3.50 GHz CPU, two Nvidia GTX 1080 Ti GPUs, 64 GB of RAM, and Ubuntu 18.04.

Convolution Stage 2				
Rank	Accuracy (%) without Fine-Tuning		Accuracy (%) with Fine-Tuning	
	NL2A [12]	This Work	NL2A [12]	This Work
3/4	92.34	**93.01**	92.78	**93.01**
1/2	92.00	**92.76**	92.44	**92.84**
1/4	90.31	**91.59**	91.56	**92.04**
Convolution Stage 3				
Rank	Accuracy (%) without Fine-Tuning		Accuracy (%) with Fine-Tuning	
	NL2A [12]	This Work	NL2A [12]	This Work
3/4	91.55	**92.85**	92.52	**92.86**
1/2	90.80	**92.15**	91.83	**92.22**
1/4	86.82	**87.88**	89.66	**90.55**
Convolution Stage 4				
Rank	Accuracy (%) without Fine-Tuning		Accuracy (%) with Fine-Tuning	
	NL2A [12]	This Work	NL2A [12]	This Work
3/4	93.07	**93.12**	93.07	**93.16**
1/2	93.11	**93.13**	93.11	**93.13**
1/4	93.09	**93.12**	**93.12**	**93.12**

Table 2.1: Single Stage Results of VGG16 on CIFAR-10

2.6.1 Performance Evaluations of Individual Convolution Stages on CIFAR-10

First, we examine the performance of our algorithm on one single convolution layer of the VGG16 network on the CIFAR-10 dataset. During the experiment, we only apply the low-rank approximation to one convolution layer and fix the other layers. We recognize that the contribution of one convolution layer in a deep neural network is very small, which is hard for us to conduct effective performance comparisons. Instead, we partition the convolution layers into multiple stages or groups. We then apply the approximate to one stage of convolution layers and examine its performance. In each stage, we perform low-rank approximation of the convolution stage at three pre-defined ranks: $m = (3/4)n$, $m = (1/2)n$, and $m = (1/4)n$, and measure the classification accuracy of the corresponding approximated network. For example, suppose that the number of filters of the second convolution stage is 128, which implies that the rank of original output matrix n is 128. After we perform low-rank approximation of this stage with the above pre-defined ranks, the number of filters of the approximated network will be 96, 64 and 32, respectively.

We train the VGG16 network on the CIFAR-10 dataset and the accuracy of this baseline model is 93.13%. Table 2.1 summarizes the classification accuracy with and without fine-tuning using the $\mathcal{L}_2$-norm low-rank approximation developed by Zhang *et al.* [12] and the linear $\mathcal{L}_1$-norm low-rank approximation developed in this work. We report the accuracy for the approximation of three convolution stages, 2, 3 and 4. We can see that our proposed method outperforms the $\mathcal{L}_2$-norm based method. We can also see that the fine-tuning can further improve the classification accuracy. We can also see that, although the complexity of the convolution layer has been reduced

Rank	VGG16	VGG19
Full Rank	8.71ms (1.00×)	11.3ms (1.00×)
1/2 Rank	4.94ms (1.76×)	7.86ms (1.43×)
1/4 Rank	3.13ms (2.78×)	6.28ms (1.79×)

Table 2.2: Execution Time on CPU of VGG Whole-Model Approximation

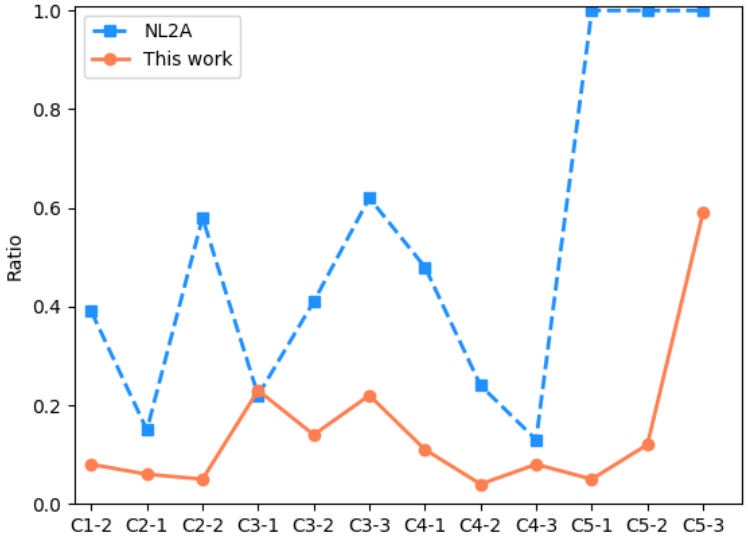
Accuracy (%)	NL2A [12]	This Work
1/2 Rank without Fine-Tuning	90.68	**92.14**
1/2 Rank with Fine-Tuning	91.47	**92.18**
1/4 Rank without Fine-Tuning	73.87	**89.81**
1/4 Rank with Fine-Tuning	88.39	**89.57**
Accuracy (%)	VH2 [15]	This Work with Iterative Two-step
1.76× Speed	88.48	**92.25**
2.78× Speed	88.38	**89.81**

Table 2.3: Results of VGG16 Whole-Model Approximation

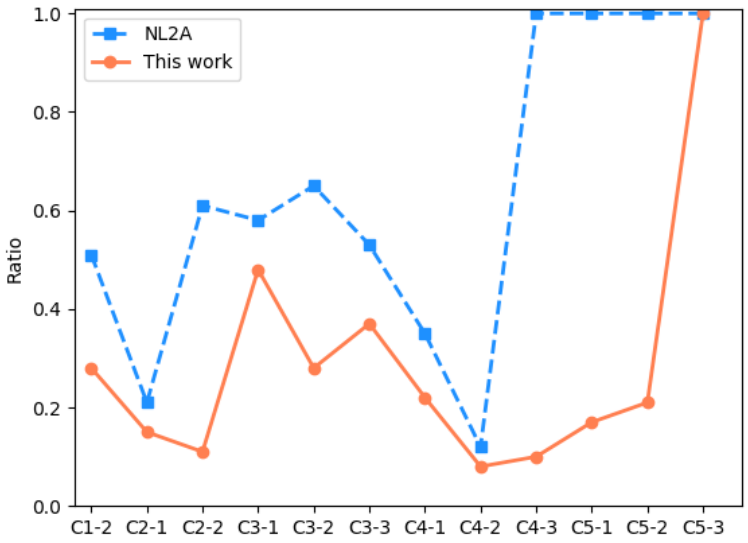
significantly, the amount of classification accuracy drop is very small, especially for stages 2 and 4. Fig. 2.4 shows the computational complexity of each convolution layer and their approximated versions at rank of 1/2 and 1/4. The top and bottom figures are for VGG16 and VGG19 networks, respectively. We can see that our proposed method is able to significantly reduce the approximation error for both networks.

2.6.2 Performance Evaluations of the Whole Network on CIFAR-10

In the following experiments, we apply the $\mathcal{L}_1$-norm low-rank approximation to all convolution layers simultaneously and evaluate its performance in terms of classifica-

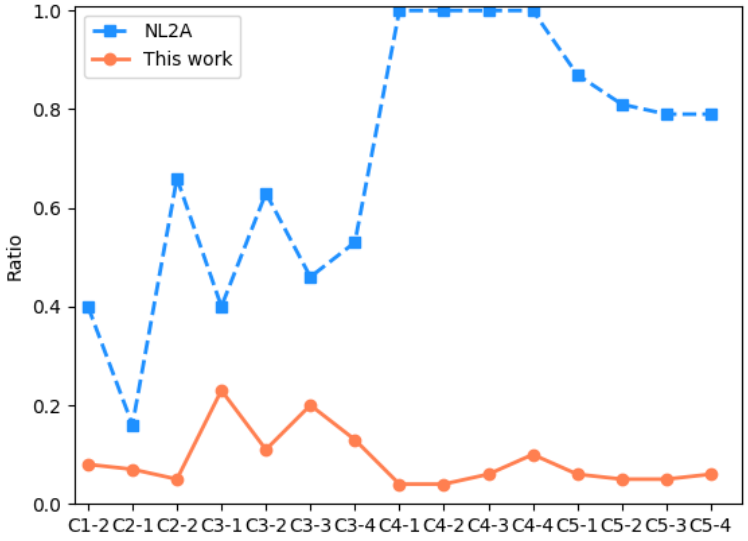


(a) VGG16 1/2 rank linear residual

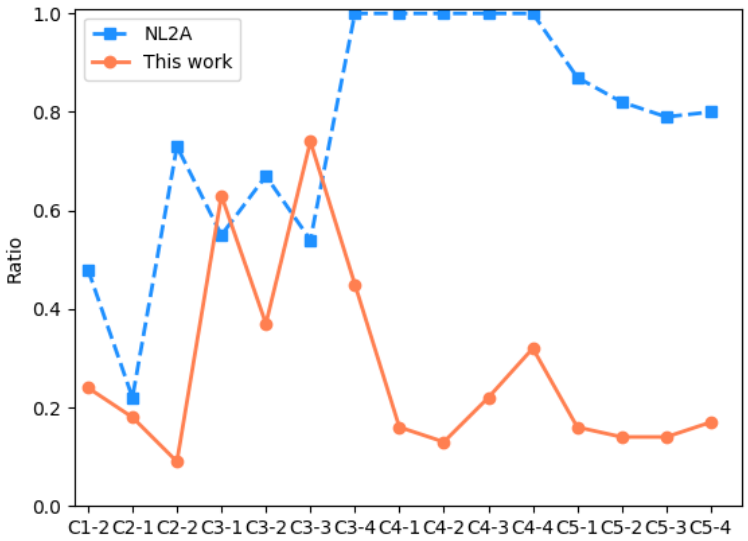


(b) VGG16 1/4 rank linear residual

Figure 2.5: Linear approximation error ratio of VGG16 at 1/2 rank and 1/4 rank.



(a) VGG19 1/2 rank linear residual



(b) VGG19 1/4 rank linear residual

Figure 2.6: Linear approximation error ratio of VGG19 at 1/2 rank and 1/4 rank.

Accuracy (%)	NL2A [12]	This Work
1/2 Rank without Fine-Tuning	87.45	**92.09**
1/2 Rank with Fine-Tuning	90.82	**92.26**
1/4 Rank without Fine-Tuning	71.60	**87.74**
1/4 Rank with Fine-Tuning	86.86	**88.86**

Accuracy (%)	VH2 [15]	This Work with Iterative Two-step
1.43× Speed	88.14	**92.34**
1.79× Speed	88.03	**89.33**

Table 2.4: Results of VGG19 Whole-Model Approximation

tion accuracy and network speed up. As we mentioned in the previous section, the baseline accuracy of the VGG16 is 93.13%. And the baseline accuracy of the VGG19 is 93.29%. We recognize that it is not efficient to apply the low-rank approximation to the first convolution layer, since the first layer deals directly with the input image. Lhe last layer in the network is also not suitable for low-rank approximation, since the correlation of features from previous deep layers is not obvious and the last layer directly linked to the final output decision. Significant rank reduction on these two layers is not cost-effective in terms of classification accuracy and network speed up. Therefore, during approximation, the first and last layers in both VGG16 and VGG19 are skipped during low-rank approximation.

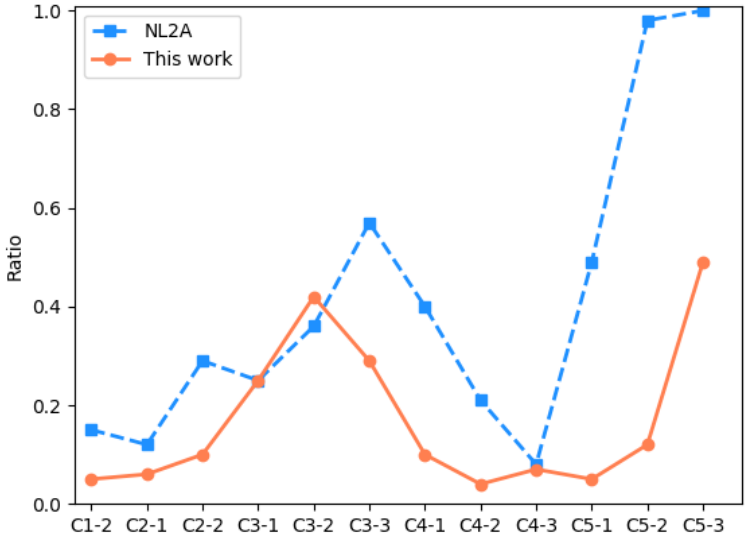
In the single convolution stage approximation, we minimize the difference between the output from non-approximated model $Y = WX$ and the output from approximated model $\hat{Y} = MWX$. In the whole-model approximation, we sequentially apply the approximation algorithm to each layer. Instead of $\hat{Y} = MWX$, we

use $\hat{Y} = MW\hat{X}$, where $\hat{X}$ is the approximation output from the previous layer. In this case, the accumulative errors in the previous layers are taken into consideration.

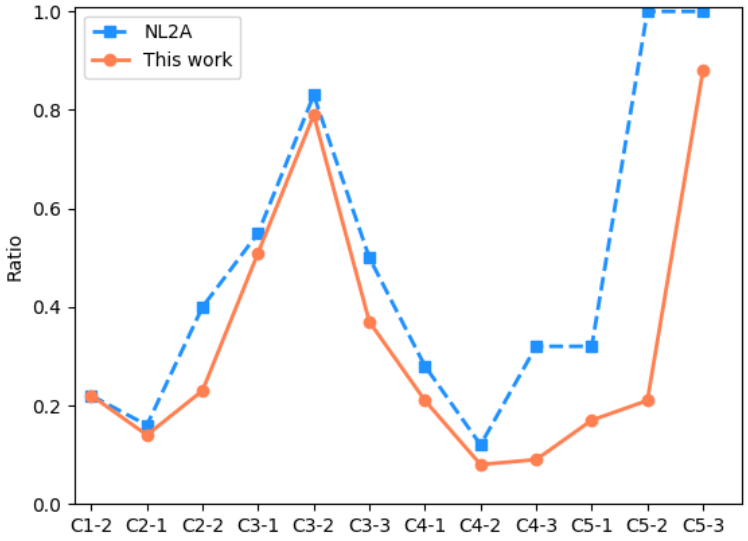
We reduce the rank from 1 to 1/2 and 1/4 at each layer and show the approximation errors measured by the Frobenius norm of the residual difference between the reconstructed output $\hat{Y}$ and the original Y without approximation. In Fig. 2.5 and 2.6, we compare the linear approximation errors ratio (*i.e.* $\|Y - \hat{Y}\|_F / \|Y\|_F$) at each convolution layer obtained by this work and NL2A [12]. Fig. 2.7 shows the non-linear approximation error ratios (*i.e.* $\|\max(Y,0) - \max(\hat{Y},0)\|_F / \|\max(Y,0)\|_F$) of the VGG16 network obtained by this work and NL2A [12]. The results for VGG19 are shown in Fig. 2.8. We can see that this work outperforms NL2A [12]. We can see that our proposed method is able to significantly reduce the approximation error in each convolution layer.

Table 2.2 shows the VGG16 and VGG19 execution time of each image on CPU, which we run the forward inference for 10 times and compute the average time. Please note that these two compared methods have the same network architecture and their complexity and cost time is exactly the same.

Table 2.3 summarizes the classification accuracy of the approximated network of VGG16 on the CIFAR-10 dataset at two different ranks, 1/2 and 1/4. We can see that the proposed method improves the classification accuracy over the NL2A method by 1% at rank 1/2. At rank 1/4, the improvement become larger, about 16% without fine-tuning. But, after fine-tuning, the difference is reduced to 1.2%. We also compare our method with iterative two-step procedure with the scheme 2 decomposition in [15] and refer it as VH2. Please note that we compare two algorithms under the same execution time. Table 2.4 summarizes the results for VGG19. Similar performance

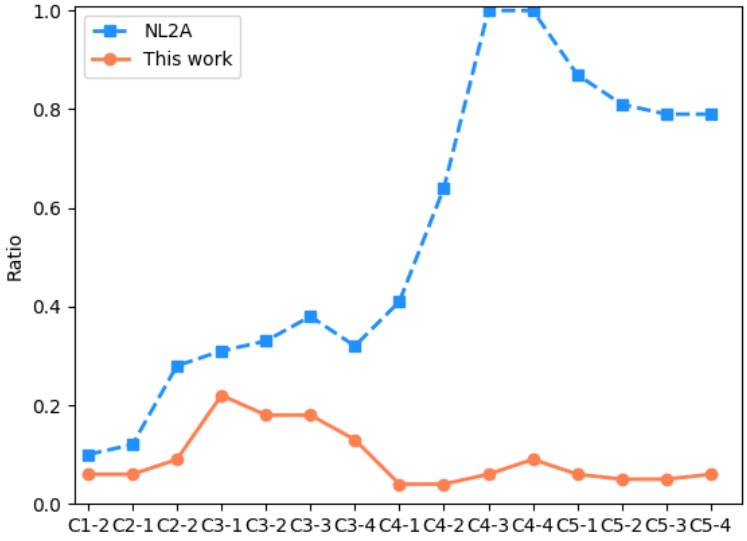


(a) VGG16 1/2 rank non-linear residual

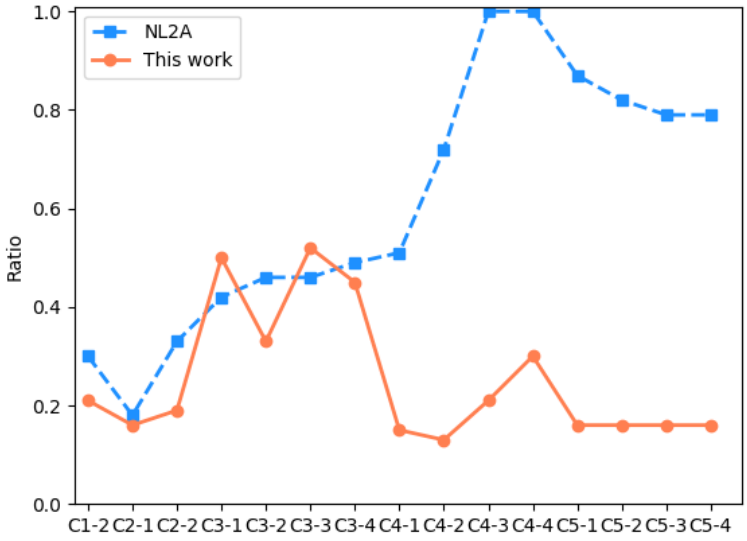


(b) VGG16 1/4 rank non-linear residual

Figure 2.7: Non-linear approximation error ratio of VGG16 at 1/2 rank and 1/4 rank.



(a) VGG19 1/2 rank non-linear residual



(b) VGG19 1/4 rank non-linear residual

Figure 2.8: Non-linear approximation error ratio of VGG19 at 1/2 rank and 1/4 rank.

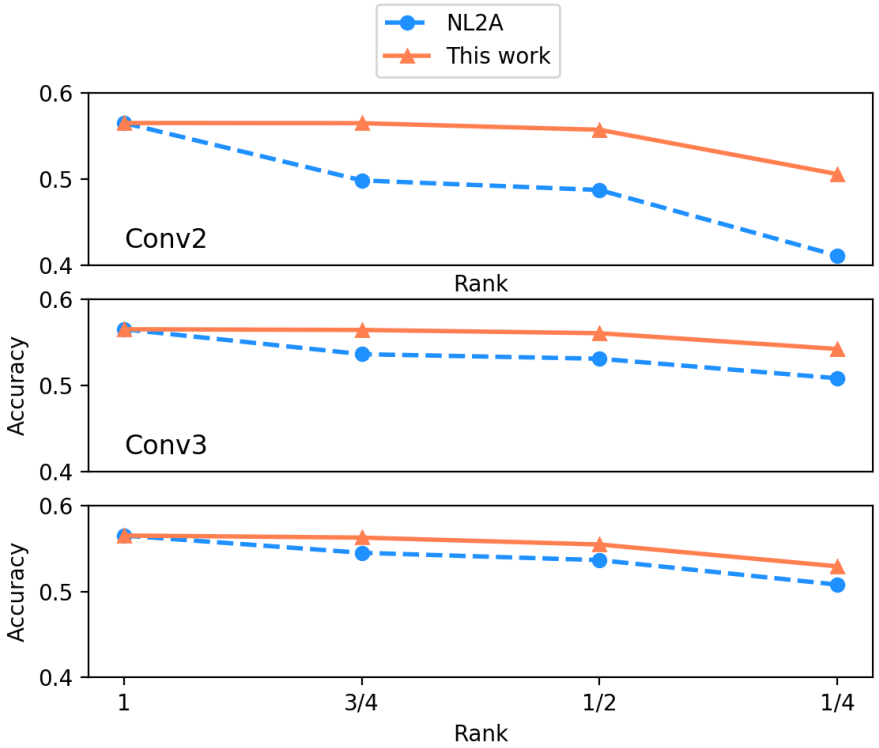


Figure 2.9: Single layer approximation accuracy on AlexNet

improvement has been achieved.

2.6.3 Performance Evaluations on the ImageNet

In this experiment, we evaluate our method on the large ILSVRC-2012 ImageNet dataset. As mentioned in the above, we recognize that a large network such as VGG16 will consume a huge amount time on this ILSVRC-2012 dataset. Instead, to demonstrate the performance of our low-rank decomposition on this large dataset, we choose the AlexNet which has relatively low computational complexity. The pre-trained baseline model is obtained from PyTorch [64] and the baseline top-1 accuracy and top-5 accuracy are 56.52% and 79.07%, respectively. During fine-tuning, input

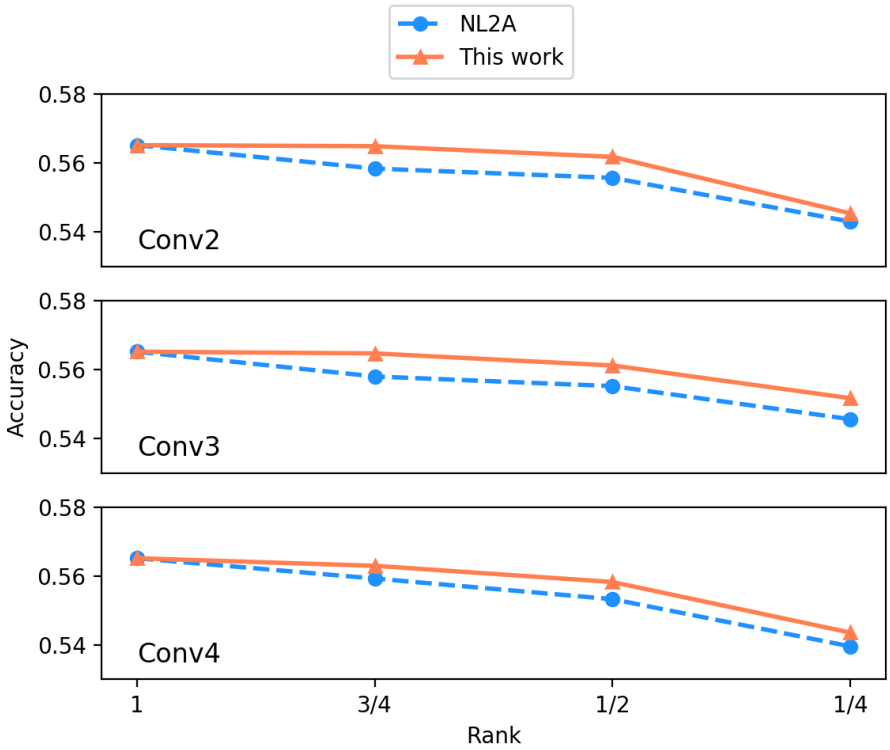


Figure 2.10: Single layer approximation accuracy with fine-tuning on AlexNet

Rank	AlexNet
Full Rank	11.04ms (1.00×)
1/2 Rank	9.08ms (1.22×)
1/4 Rank	6.90ms (1.60×)

Table 2.5: Execution Time on CPU of AlexNet Whole-Model Approximation

AlexNet	Layer	Complexity (FLOPs)		
		Original	Approximated	Remain Ratio
1/2 Rank	C1	7.03×10^7	4.13×10^7	0.59
	C2	2.24×10^8	1.25×10^8	0.56
	C3	1.21×10^8	6.85×10^7	0.57
	C4	1.50×10^8	8.03×10^7	0.53
	C5	9.97×10^7	5.54×10^7	0.56
1/4 Rank	C1	7.03×10^7	2.07×10^7	0.29
	C2	2.24×10^8	6.27×10^7	0.28
	C3	1.21×10^8	3.43×10^7	0.28
	C4	1.50×10^8	4.01×10^7	0.27
	C5	9.97×10^7	2.77×10^7	0.28

Table 2.6: Remained FLOPs of AlexNet

images are resized to 256×256. The resized images are randomly cropped to 224×224 images and randomly horizontally flipped for data argumentation. During network inference, images are center cropped.

Fig. 2.9 shows the accuracy of the network when the low-rank approximation is applied to one layer without fine-tuning on layers *C2*, *C3* and *C4*. Fig. 2.10 shows the accuracy with fine tuning. Table 2.6 and 2.7 shows the original computational complexity (in FLOPS) of the network in Column 3, the reduced complexity after low-rank approximation by our algorithm in Column 4, and the complexity reduction ratio in Column 5. We can see that the final complexity is very close to the target rank. In Column 6 and 7, the table shows the linear approximation error ratio of each layer when our algorithm and the NL2A method are applied, respectively. We can see that

AlexNet	Layer	Linear Approximation Error Ratio		Non-linear Approximation Error Ratio	
		NL2A [12]	This Work	NL2A [12]	This Work
1/2 Rank	C1	0.63	**0.04**	0.23	**0.08**
	C2	0.57	**0.16**	0.48	**0.28**
	C3	0.51	**0.19**	0.43	**0.33**
	C4	0.65	**0.32**	0.56	**0.46**
	C5	0.67	**0.42**	0.51	**0.42**
1/4 Rank	C1	0.67	**0.09**	0.43	**0.21**
	C2	0.69	**0.37**	**0.62**	0.66
	C3	0.57	**0.38**	**0.51**	0.58
	C4	0.74	**0.50**	**0.68**	0.73
	C5	0.70	**0.69**	**0.65**	0.81

Table 2.7: Linear Approximation Error Ratio of AlexNet Whole Network Approximation

our method is able to significantly reduce the approximation error. Table 2.8 shows the accuracy of the AlexNet after the whole network has been approximated using low-rank approximation at ranks of 1/2 and 1/4. We can see that our method outperforms the NL2A method by up to 3.1%. Note that, at rank 1/4, our method performs worse than the NL2A without fine-tuning. But, after fine-tuning or iterative two-step procedure with fine-tuning, our method can be significantly improved, outperforming the NL2A method when the same fine-tuning is applied. The execution time is shown in Table 2.5

Accuracy (%) (Top 1/Top 5)	NL2A [12]	This Work
1/2 Rank without Fine-Tuning	50.28/74.21	**53.68/77.30**
1/2 Rank With Fine-Tuning	53.19/76.78	**54.86/78.03**
1/4 Rank without Fine-Tuning	**41.28/65.63**	36.40/61.06
1/4 Rank with Fine-Tuning	48.18/72.35	**48.88/72.94**

Accuracy (%)	VH2 [15]	This Work with Iterative Two-step
1.22× Speed	53.21/76.41	**55.19/78.37**
1.60× Speed	50.23/74.09	**50.92/74.56**

Table 2.8: Results of AlexNet Whole-Model Approximation

Chapter 3

Adversarial Attack Noise Removal Based on Low-Rank Completion of High-Sensitivity Points

3.1 Motivation

Existing noise removal-based methods suffer from performance degradation due to damages on the original images. In order to remove the sophisticated attack noise, the noise removal algorithm often applies heavy smoothing operations to the whole image and cause significant damages to the non-attacked areas.

In this work, we observe that the adversarial attack noise is not uniformly distributed over the image. Attack noise at different image locations will have different impact on the network prediction output.

Fig. 3.1 shows two examples of clean images of digits and their attacked version. For example, the clean image of digit 5 in (a) is being attacked and becomes the

(a) Clean digit 5

(b) Adversarial digit 5

(c) Clean digit 8

(d) Adversarial digit 8

Figure 3.1: Clean digit images and adversarial digit images.

image in (b). The clean image of digit 8 is attacked and becomes the image in (d). We can see that those pixels that fills the gap in 5 cause the mis-classification. Also, in (d), the attacker removes a group of pixels on the right edge of 8 and cause the mis-classification into 6. This example demonstrates that different attack noise pixels

have different impact on the network classification performance. In other words, some image pixel locations are sensitive for image classification while other pixels are less sensitive.

Based on this observation, in this work, we propose to develop a new approach to defending image against adversarial noise. As illustrated in Fig. 3.2, we first develop an approach to detect the high-sensitivity pixel locations. Attacks to these pixels will have larger impact on the network prediction performance (e.g. classification accuracy) than other low-sensitivity pixels. For the high-sensitivity pixels, we develop a low-rank structure-preserving image completion algorithm to remove the attack noise and restore the image. For other image regions, i.e., the low-sensitivity pixels, we use the image de-noising method based on TV (total variation) norm minimization [27]. Our extensive experimental resutls on benchmark datasets demonstrate that our approach can achieve highly effective defense and outperforms existing noise removal-based defense methods with powerful black-box, gray-box, and white-box attacks.

Major Contributions The major contributions of this work can be demonstrated as follows:

- We develop a new approach for defending image against adversarial attacks based on high-sensitivity pixels selection and low-rank image completion.
- We develop a high-sensitivity pixels detection and selection methods based on back-propagated gradient information.
- We develop a structure-preserving image completion algorithm based on reweighted low-rank matrix recovery algorithm.
- We experimentally demonstrate our method can provide a more effective and

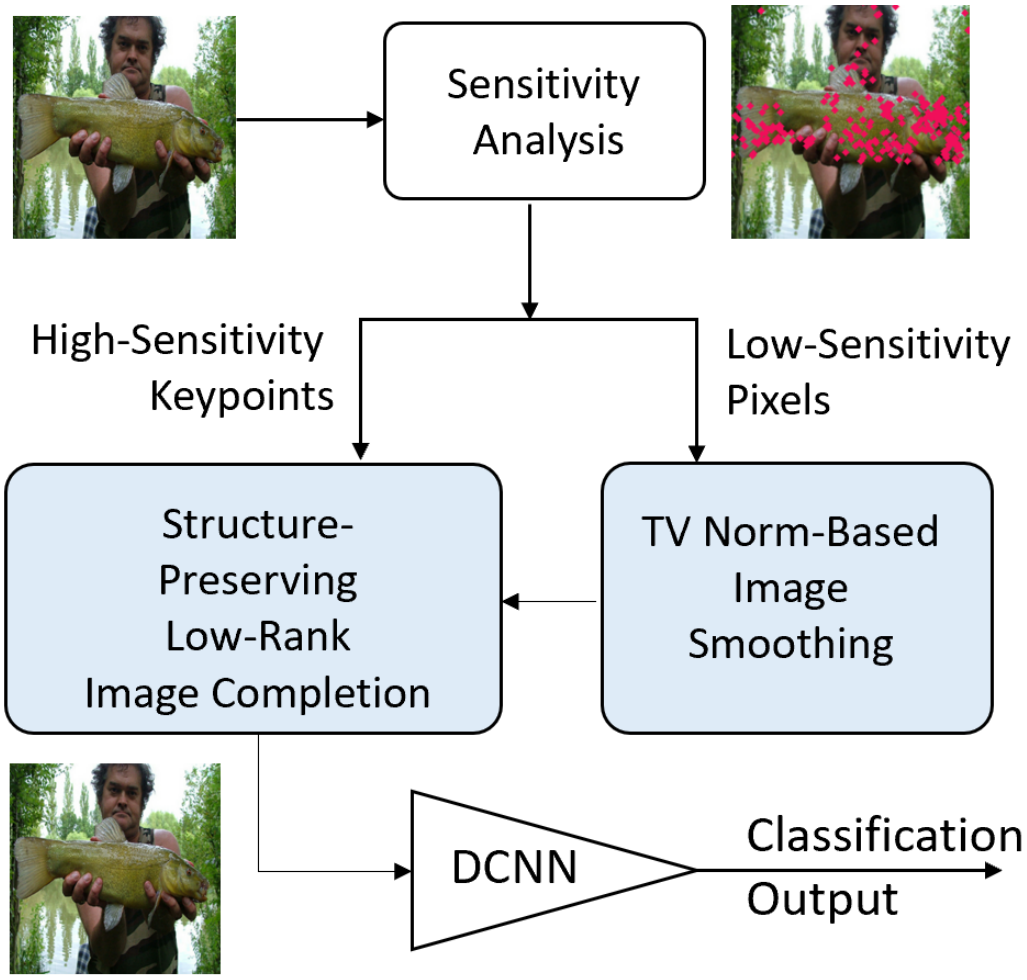


Figure 3.2: Illustration of our defense method.

robust defense than other image de-noising approaches because of the high-sensitivity points information from the protected classifier.

3.2 Related Work

This is work is closely related to adversarial attacks of deep neural networks, defense of deep neural networks, and image / matrix completion. In this section, we review

existing work on these three topics.

3.2.1 Adversarial Attacks of Deep Neural Networks

There are two major approaches in generating adversarial attacks for deep neural networks: *signed gradient-based methods* [20, 21, 28], and *optimization-based methods* [65, 66]. The fast gradient sign method (FGSM) is the very first singed gradient-based attack method proposed by Goodfellow *et al.* [20]. This method simulates the network training process, assigns a wrong label to the input image, then back propagates the error gradients through the network layers all the way to the input image $\mathbf{X}$. Specifically, let y be the wrong label, it generates the perturbation $\mathbf{\Delta}$ by simply taking the sign of the gradient for at each image pixel:

$$\mathbf{X^a} = \mathbf{X} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{X}} L(\mathbf{X}, y)), \tag{3.1}$$

where, ϵ is the perturbation magnitude and $L(\mathbf{X}, y)$ is the cross-entropy loss. The basic iterative method (BIM) [21] is an iterative attack method based on the FGSM. It is able to generate very strong image perturbation:

$$\mathbf{X^{a,n}} = \mathbf{X^{a,n-1}} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{X^{a,n-1}}} L(\mathbf{X^{a,n-1}}, y)), \tag{3.2}$$

where $n = 1, ..., N$, N is the maximum iteration number and $\mathbf{X^{a0}} = \mathbf{X}$ which is the original image. Athalye *et al.* [28] observed that current defense algorithms, except adversarial training, are essentially masking gradients and developed a state-of-the-art signed gradient-based Backward Pass Differentiable Approximation (BPDA)

algorithm for attack:

$$\mathbf{X^{a,n}} = \mathbf{X^{a,n-1}} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{X^{a,n-1}}} L(\mathbf{DEF}(\mathbf{X^{a,n-1}}), y)), \tag{3.3}$$

where $\mathbf{DEF}(\cdot)$ is the function of defense algorithms. It should be noted that this BPDA algorithm only works under the white-box mode with full knowledge of and access to the defense method. It has been recognized that fast-gradient based attack methods often generate large perturbations, easily seen with human eyes. *DeepFool* [65] aims to minimize the image perturbation. It finds the label with the minimal distance $\hat{l}(\mathbf{X})$ between the input and all classification boundaries except the true label $\hat{k}(\mathbf{X})$ to generate the perturbation. This procedure is repeated until the classifier yields the wrong result.

3.2.2 Defense Methods for Deep Neural Networks

As a counterpart to the adversarial attack methods, defense methods aim to defend the neural networks against adversarial attacks by removing attack noise from images, improving the capability of the network to handle adversarial attacks. Various approaches have been developed for network defense, including image de-noising [24] and adversarial training [67, 68, 69]. Image de-noising algorithms pre-process the input images to eliminate perturbations before forwarding the image to the classifier. Adversarial training algorithms aim to train a robust classifier to resist attacks. Our proposed method falls into the category of image de-noising.

Guo *et al.* [24] proposed two image transformations, *total variance minimization* and *image quilting*, to remove attack noise for gray-box and black-box attacks. Mag-

Net [70] is proposed to detect the perturbations and then reshape them according to the difference between clean and adversarial images. Jia *et al.* [71] defend adversarial examples by an image compression framework *ComDefend*. Xie *et al.* [72] introduced the feature de-noising module in the intermediate layer for defending strong PGD white-box attacks. Networks path is considered as a useful tool for adversarial images detection in [73, 74]. Buckman *et al.* [75] proposed a method using thermometer encoding to defend against the adversarial images. Recently, some methods have been developed to provide certification for the classifier to protect the classifier from any adversarial perturbation within a certain range [76, 77, 78, 79].

Adversarial training [20] aims to improve the robustness of classifiers by training the network with adversarial images. PGD attack (*i.e.* BIM attack with random initial noise) is suggested for adversarial training in [67] due to its strong perturbation. Kannan *et al.* [68] adds a regularization logits pair term in the training loss, which aims to minimize the logits difference between the clean image and the attacked image. An ensemble adversarial training in [69] is proposed to resist all kinds of attacks, and is especially effective in black-box defenses. Recently, some works explore the deep generative models, such as GANs, to recovery the clean image from adversarial attack. Samangouei *et al.* [26] clean the adversarial images by a trained generative adversarial network (GAN) with multiple iterations. Our proposed method can be coupled with adversarial training to further improve the defense performance.

3.2.3 Low-Rank Completion of Matrices and Images

Matrix completion [80, 81, 82, 83, 84] aims to reconstruct a large low-rank matrix with only a small number of known entries. This has been a critical study in many

areas of science and engineering application such as computer vision, recommendation systems, sparse coding, and machine learning [82, 84]. Let matrix $\mathcal{M}$ be the input image with rank r, and only ω entries can be observed. The matrix completion task is to estimate the unknown entries in matrix $\mathcal{M}$. The problem can be formulated as follows:

$$\min_X rank(X), \quad s.t. \quad X_{i,j} = \mathcal{M}_{i,j}, \quad (i,j) \in \Omega, \tag{3.4}$$

where X is the recovered matrix and Ω is the set of pixel locations with observed values in $\mathcal{M}$. The objective $rank(X)$ is used to achieve a low-rank structure of matrix X. However, the optimization problem in (3.4) is NP-hard [80] because of the non-convexity of the rank function. Usually, this can be solved by replacing with the nuclear norm [85]. With the nuclear norm, the above optimization problem can be relaxed to the following convex problem:

$$\min_X \|X\|_* \quad s.t. \quad X_{i,j} = \mathcal{M}_{i,j}, \quad (i,j) \in \Omega, \tag{3.5}$$

where $\|X\|_*$ is the nuclear norm of matrix X, and it equals the sum of singular values of matrix X [83].

3.3 Detection of High-Sensitivity Keypoints

In this section, we will present our method to defend against adversarial attacks based on low-rank completion of high-sensitivity points (LRC-HSP).

By *high-sensitivity*, we mean that if they are attacked, the image analysis performance will be severely affected, for example, the classification score will be decreased

significantly. Or, for attacked images, if we successfully repair these high-sensitivity keypoints, the image classification score will be improved significantly.

In the following, we use an example to explain our proposed method for high-sensitivity keypoint detection. Let us consider one network perceptron. Let $\vec{w}$ be the network weights at this layer, $\vec{x} \in [0, 1]$ and $y \in \{0, 1\}$ are the input and the output, respectively:

$$y(\vec{x}) = \sigma(\vec{w}^T \vec{x}) \tag{3.6}$$

where $\sigma(\cdot)$ is the activation function. In a binary classification task, when the binary cross-entropy

$$L(y(\vec{x}), t) = -y(\vec{x}) \cdot \log t - [1 - y(\vec{x})] \cdot \log(1 - t) \tag{3.7}$$

is used as the loss function, the network update its weight $\vec{w}$ using the gradient decent algorithm based on the gradient of $L(y(\vec{x}), t)$ with respect to the network weight $\vec{w}$:

$$\vec{w}_n = \vec{w}_{n-1} - \alpha \nabla_{\vec{w}} L(y(\vec{x}), t), \tag{3.8}$$

where α is the learning rate. However, during adversarial attacks of images, the attacker will modify the image $\vec{x}$ instead of the network weight $\vec{w}$. Therefore, the attacker computes the gradient of the loss function $L(y(\vec{x}), t)$ with respect to the input image $\vec{x}$ and modifies the image as

$$\vec{x}' = \vec{x} - \epsilon \cdot \nabla_{\vec{x}} L(y(\vec{x}), t'), \tag{3.9}$$

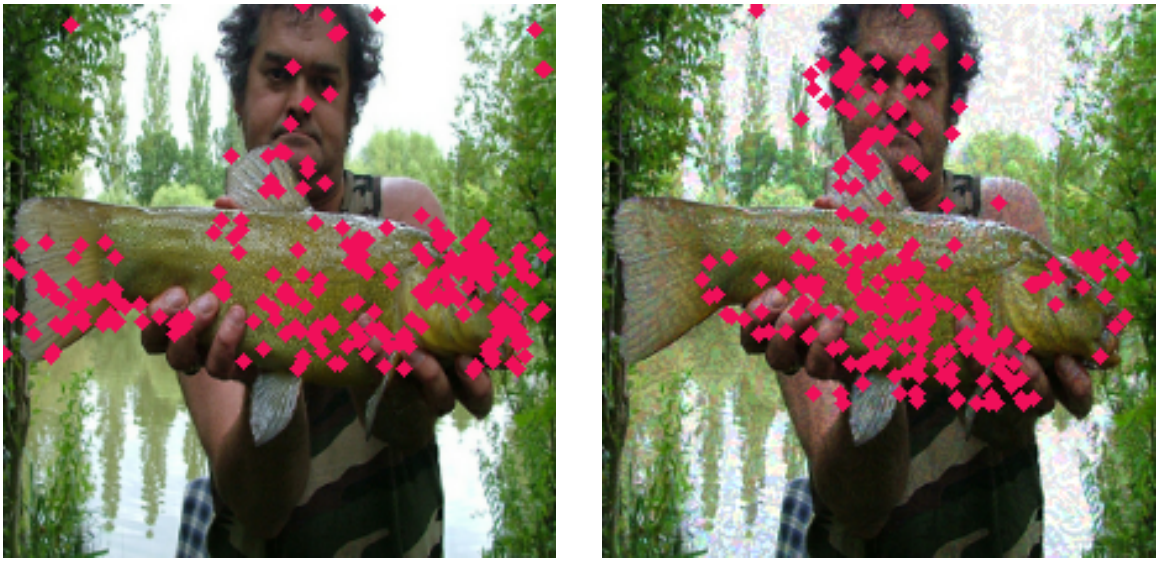
where t' is the wrong label that the attacker aims to achieve, ϵ controls the magnitude of the adversarial attack noise. This formula suggests that the attacker wishes to

modify a pixel of the image based on the value of $\nabla_{\vec{x}} L(y(\vec{x}), t')$ so as to achieve its objective of falsifying the network. In other words, if the value $\nabla_{\vec{x}} L(y(\vec{x}), t')$ at pixel x_a is larger than the value at pixel x_b, then pixel x_a is more important or more sensitive than x_b from the network performance perspective. In this case, we call pixel x_a has higher sensitivity than pixel x_b. Based on this observation, we propose to classify the image pixels into two categories: high-sensitivity keypoints and low-sensitivity pixels based on the following criteria

$$\mathbf{M}(\vec{x}) = \begin{cases} 1, & ||\nabla_{\vec{x}} L(y(\vec{x}), t')|| > \alpha_0 \\ 0, & otherwise. \end{cases} \tag{3.10}$$

Here, $\mathbf{M}(\vec{x})$ defines a binary mask which records the positions of all high-sensitivity keypoints. α_0 is a threshold. In this work, we choose the top certain percentage of pixels, for example, top 5% or 10% of pixels, as high-sensitivity keypoints. In this case, α_0 is determined by this percentage.

Figure 3.3 shows the masked clean image and masked adversarial image by their $\mathbf{B}_{more}$ and $\mathbf{B}_{less}$ respectively. The true label of this image is tench, and the adversarial images generated by FGSM is classified as vestment. The more-related points on the clean image are distributed on the fish, while the more-related points of the adversarial image are distributed not only on the fish and but also on the human face. In the case of less-related points on the fish, the number of the points on adversarial images are larger than the number of the points on the clean images, especially on the edge.



(a) Clean images (b) Adversarial images

Figure 3.3: Masked images (3% points removal): (a) the more-related points on clean image, (b) the more-related points on adversarial image.

3.4 Structure-Preserving Image Completion at Detected High-Sensitivity points

After locating the high-sensitivity keypoints, we will proceed to remove the attack noise and recover the image. Specifically, in the low-sensitivity regions, we apply existing noise removal and image smoothing algorithm to remove the adversarial attack noise since these low-sensitivity pixels do not contribute as significantly as those high-sensitivity keypoints to the overall image classification performance. In this work, we use the image smoothing algorithm based on total variation (TV) norm developed in [27] due to its capability of removing noise while preserving local image structures. For the high-sensitivity keypoints, we found out that it is not efficient to apply the same TV norm-based noise removal algorithm. Instead, we propose to develop the following structure-preserving image completion algorithm.

3.4.1 Structure-Preserving Image Completion

The task of matrix or image completion is to fill in missing elements of the matrix or missing pixels of the image by exploring their low-rank structure [82, 84]. In this work, we recognize that it is not efficient to apply this low-rank assumption to high-sensitivity image points since they often contain significant amounts of distinctive structure information for image classification or object detection. To address this issue, we propose to incorporate the weighted nuclear norm into the image completion analysis. In our previous work [86], we have demonstrated that the nuclear norm is very effective in low-rank image restoration and noise removal. In this work, we propose to extend this analysis to low-rank image completion for removing high-sensitivity adversarial noise points. Let $X^a \in R^{m \times n}$ of size $m \times n$ be the image which has been attacked with adversarial noise. Let $\mathbf{M}$ be the binary mask which indicates the locations of high-sensitivity points. If the pixel at location(i, j) is selected as the high-sensitivity point, then $\mathbf{M}(i, j) = 1$, otherwise $\mathbf{M}(i, j) = 0$ Then the matrix completion problem can be formulated as:

$$\min_X \sum_{j=1}^{n} w_j^X \cdot \widetilde{\sigma}_j, s.t. \quad X_{i,j} = X_{i,j}^a \; for \; \mathbf{M}(i, j) = 0, \tag{3.11}$$

where $W^X = \{w_j^X\}$ are weights for $\{\widetilde{\sigma}_j\}$. $\{\widetilde{\sigma}_j\}$ are singular values of matrix X. $\sum_{j=1}^{n} w_j^X \cdot \widetilde{\sigma}_j$ represents the weighted nuclear norm of X. To enhance the structural smoothness and separating the low-rank component from the noisy image simultaneously, in our previous work [86], we incorporated the TV regularization into the re-weighted low-rank matrix decomposition framework. It should be noted that this method was developed for the low-rank decomposition of the whole image or ma-

trix. In this paper, we propose to extend this method to image completion at the high-sensitivity keypoints, which is a different and new problem.

Specifically, we incorporate the TV regularization into the low-rank matrix completion to ensure structural smoothness in the result image. We rewrite the constraint $X_{i,j} = X^a{}_{i,j}$ for $\mathbf{M}(i,j) = 0$ as a penalty term based on the following $\mathcal{L}_1$ norm of weighted errors between the attacked image X^a and the original image X:

$$\vec{e}_M = \|W^e \odot \hat{\mathbf{M}} \odot E\|_1, \quad \hat{\mathbf{M}} = \mathbf{I} - \mathbf{M}, \quad E = X^a - X, \tag{3.12}$$

where $\odot$ represents element-wise matrix multiplication, $\mathbf{M}$ is the binary mask representing the locations of low-sensitivity pixels. $\mathbf{I}$ is the matrix with all elements as one. The matrix $W^e = \{w^e_{ij}\} \in R^{m \times n}$ is the weights for the image completion errors at each pixel location. With this, the low-rank image completion problem can be formulated as

$$\begin{aligned} & \min_{X,E} \sum_{j=1}^{n} w_j^X \cdot \widetilde{\sigma}_j + \lambda \|W^e \odot \hat{\mathbf{M}} \odot E\|_1 + \eta \|X\|_{TV}, \\ s.t. \quad & \Delta_L \leq x_{i,j} \leq \Delta_U, \\ & X^a - X = E, \end{aligned} \tag{3.13}$$

where $W^X = \{w_j^X\}$ is the weight for $\{\widetilde{\sigma}_j\}$. If the weight w^e_{ij} is set as the inverse of the absolute value of $E(i,j)$, i.e., $w^e_{ij} = 1/|E(i,j)|$ (suppose $E(i,j) \neq 0$), we have $\|W^e \odot E\|_1 = \|E\|_0$, where the $\mathcal{L}_0$-norm $\|\cdot\|_0$ is the number of nonzero entries in matrix E. In our experiments, we find that the weighted sparse error matrix $\|W^e \odot E\|_1$ can better approximate $\|E\|_0$ than $\|E\|_1$. This is helpful for enhancing the sparsity of the error matrix E. The constraint Δ_L and Δ_U represent that the lower and upper

bounds of each pixel. For example, in an image, the pixel value has a range of $[0, 255]$.

Using the above procedure, we have converted the matrix completion problem in (3.11) into a matrix decomposition problem when the constraint $X_{i,j} = X^a{}_{i,j}$ for low sensitivity pixels with $\mathbf{M}(i,j) = 0$ is replaced by a $\mathcal{L}_0$-norm of weighted errors. We will use augmented Lagrange method to solve this low-rank matrix decomposition problem. To simplify the optimization problem, we introduce a new auxiliary variable Z to the model as follows:

$$\begin{aligned} \min_{Z,X,E} \quad & \sum_{j=1}^{n} w_j^Z \cdot \sigma_j + \eta \|X\|_{TV} + \lambda \|W^e \odot \hat{\mathbf{M}} \odot E\|_1 \\ s.t. \quad & \Delta_L \leq x_{i,j} \leq \Delta_U, \\ & X^a - Z = E, Z = X, \end{aligned} \tag{3.14}$$

where $W^Z = \{w_j^Z\}$ are the weights for $\{\sigma_j\}$, $\{\sigma_j\}$ are the singular values of matrix Z, and $w_j^Z = w_j^X, j = 1, \cdots, n$. Thus, the augmented Lagrangian function of (3.14) is constructed as:

$$\begin{aligned} & f(Z, X, E, Y_1, Y_2) \\ = \quad & \sum_{j=1}^{n} w_j^Z \cdot \sigma_j + \eta \|X\|_{TV} + \lambda \|W^e \odot \hat{\mathbf{M}} \odot E\|_1 \\ + \quad & \langle Y_1, X^a - Z - E \rangle + \langle Y_2, X - Z \rangle \\ + \quad & \frac{\mu}{2}(\|X^a - Z - E\|_F^2 + \|X - Z\|_F^2), \\ s.t. \quad & \Delta_L \leq x_{i,j} \leq \Delta_U, \end{aligned} \tag{3.15}$$

where $\langle \cdot, \cdot \rangle$ is the inner product of two matrices. To simplify the objective function, we consider W^Z and W^e as constants.

3.4.2 An Iterative Solution

Different from the image restoration problem in [83], the above optimization problem for structure-preserving low-rank image completion is new and challenging. This requires us to derive a new mathematical analysis and solution. We solve this problem by the iterative alternating direction method (ADM). By optimizing one variable while fixing the other variables in an iterative manner, the original complicated multi-variable optimization problem can be converted to several simple single-variable optimization problems. In this way, solutions can be obtained analytically. There are five major sets of variables (Z, X, E, Y_1, Y_2) in this problem. We will show how each of these variables can be optimized in the following.

(1) ***Optimizing the auxiliary varilable*** Z. Z can be optimized by minimizing $f(Z, X, E, Y_1, Y_2)$ with respect to Z while fixing variables (X, E, Y_1, Y_2). Specifically,

$$\begin{aligned}
& \arg\min_{Z} f(Z, X, E, Y_1, Y_2) \\
\overset{a}{=} & \arg\min_{Z} \sum_{j=1}^{n} w_j^Z \cdot \sigma_j \\
& + \langle Y_1, X^a - Z - E \rangle + \langle Y_2, X - Z \rangle \\
& + \frac{\mu}{2}(\|X^a - Z - E\|_F^2 + \|X - Z\|_F^2) \\
\overset{b}{=} & \arg\min_{Z} \sum_{j=1}^{n} w_j^Z \cdot \sigma_j \\
& + \mu \|Z - \frac{1}{2}(X^a + X - E + \frac{Y_1}{\mu} + \frac{Y_2}{\mu})\|_F^2 \\
\overset{c}{=} & \arg\min_{Z} \sum_{j=1}^{n} w_j^Z \cdot \sigma_j + \mu \|Z - L\|_F^2, \qquad (3.16)
\end{aligned}$$

where

$$L = \frac{1}{2}(X^a + X - E + \frac{Y_1}{\mu} + \frac{Y_2}{\mu}). \tag{3.17}$$

The equality $\stackrel{a}{=}$ of (3.16) is obtained by reducing the items without Z, and equality $\stackrel{b}{=}$ and $\stackrel{c}{=}$ are obtained by expanding the squared item of the Frobenius norm and reconstructing the new squared item of the Frobenius norm. Based on this conversion, the minimization problem in (3.16) can be solved by the non-uniform singular value thresholding (NSVT) method [87], whose solution is given by :

$$Z = \mathcal{D}_{(2\mu)^{-1}W^Z}(L), \tag{3.18}$$

where $\mathcal{D}$ is the non-uniform singular value thresholding operator.

(2) ***Optimizing the low-rank and smoothness varilable*** X. If variables (Z, E, Y_1, Y_2) are fixed, then X can be optimized by minimizing $f(Z, X, E, Y_1, Y_2)$ with respect to X. Specifically,

$$\begin{aligned}
& \arg\min_X f(Z, X, E, Y_1, Y_2) \\
\stackrel{a}{=} & \arg\min_X \eta\|X\|_{TV} + \langle Y_2, X - Z\rangle + \frac{\mu}{2}\|X - Z\|_F^2 \\
\stackrel{b}{=} & \arg\min_X \eta\|X\|_{TV} + \frac{\mu}{2}\|X - (Z - \frac{Y_2}{\mu})\|_F^2 \\
= & \arg\min_X \eta\|X\|_{TV} + \frac{\mu}{2}\|X - R\|_F^2,
\end{aligned} \tag{3.19}$$

where $\Delta_L \leq x_{i,j} \leq \Delta_U$ and

$$R = (Z - \frac{Y_2}{\mu}). \tag{3.20}$$

The equality $\stackrel{a}{=}$ of (3.19) is obtained by reducing the items without X, and equality $\stackrel{b}{=}$ is obtained by expanding and reconstructing the squared item of the Frobenius

norm. This minimization problem for X in (3.19) can be solved by the *Fast Gradient Projection* (FGP) algorithm [27].

Algorithm 2 ADM Algorithm for computing the Smoothed Low-Rank Matrix X

Require: Data matrix $X^a \in R^{m\times n}$, $\hat{\mathbf{M}} \in R^{m\times n}$, $W^X \in R^{1\times n}$, $W^e \in R^{m\times n}$, λ, η and δ.

Ensure: Initialize $W^Z = W^X$, $X_0 \in R^{m\times n}$, $E_0 \in R^{m\times n}$, $Z_0 \in R^{m\times n}$, $Y_{1,0} \in R^{m\times n}$, $Y_{2,0} \in R^{m\times n}$, $\mu_0 > 0$, $\xi = 10^{-7}$, $t = 0$ and $inneriter = 100$.

while $\|X^a - Z_t - E_t\|_F / \|X^a\|_F > \xi$ and $t < inneriter$ **do**

step 1: Let $L_{t+1} = X^a + X_t - E_t + Y_{1t}/\mu_t + Y_{2t}/\mu_t$, then, $Z_{t+1} = \mathcal{D}_{\mu_t^{-1}W^Z}(L_{t+1})$;

step 2: Let $R_{t+1} = Z_{t+1} - Y_{2,t}$, $\rho = \eta/\mu_t$; using the **FGP Algorithm** [27] to compute X_{t+1} based on (3.19) ;

step 3: $E_{t+1} = \mathcal{S}_{\lambda\mu_t^{-1}W^e\odot\hat{\mathbf{M}}}[X^a - Z_{t+1} + Y_{1,t}/\mu_t]$;

step 4: $Y_{1,t+1} = Y_{1,t} + \mu_t(X^a - Z_{t+1} - E_{t+1})$;

step 5: $Y_{2,t+1} = Y_{2,t} + \mu_t(X_{t+1} - Z_{t+1})$;

step 6: $\mu_{t+1} = \delta\mu_t$, $t \leftarrow t+1$;

end while

Output X^*.

(3) ***Optimizing the sparse error matrix variable*** E. We discuss how to optimize E while the other variables (Z, X, Y_1, Y_2) are fixed. This can be achieved by minimizing $f(Z, X, E, Y_1, Y_2)$ with respect to E. Specifically,

$$\begin{aligned}
& \arg\min_E f(Z, X, E, Y_1, Y_2) \\
\stackrel{a}{=} \; & \arg\min_E \lambda\|W^e \odot \hat{\mathbf{M}} \odot E\|_1 \\
& + \langle Y_1, X^a - Z - E\rangle + \frac{\mu}{2}\|X^a - Z - E\|_F^2 \\
\stackrel{b}{=} \; & \arg\min_E \lambda\|W^e \odot \hat{\mathbf{M}} \odot E\|_1 \\
& + \frac{\mu}{2}\|E - (X^a - Z + \frac{Y_1}{\mu})\|_F^2. \qquad (3.21)
\end{aligned}$$

Similar with above method, the equality $\stackrel{a}{=}$ of (3.21) is also obtained by reducing the items without E, and equality $\stackrel{b}{=}$ is obtained by expanding and reconstructing the

squared item of the Frobenius norm. This minimization problem can be solved using the non-uniform soft thresholding (NST) [88] as follows:

$$E = \mathcal{S}_{\lambda\mu^{-1}W^e \odot \hat{\mathbf{M}}}[X^a - Z + \frac{Y_1}{\mu}]. \tag{3.22}$$

where $\mathcal{S}$ is the non-uniform soft thresholding opterator. Y_1 and Y_2 are the Lagrange multiplier matrices of the original optimization problem. They should be updated when the other variables have been updated. Y_1 can be updated as follows if the other variables are fixed:

$$Y_1 = Y_1 + \mu(X^a - Z - E). \tag{3.23}$$

Similarly, Y_2 can be updated as follows if the other variables are fixed:

$$Y_2 = Y_2 + \mu(X - Z). \tag{3.24}$$

Algorithm 3 Reweighted Low-rank Keypoints Completion

Require: $X^a \in R^{m\times n}$, $\hat{\mathbf{M}} \in R^{m\times n}$, l, u, λ, η, and δ.
Ensure: initialize $W^X = (w_j^X) \in R^m$, $W^e = (w_{ij}^e) \in R^{m\times n}$, $k = 0$, $X_0 = O_{m\times n}$, $E_0 = O_{m\times n}$, $maxiter = 3$, $\mu = 10^{-6}$, $\mu_{max} = 10^{10}$.
Using IALM [89] to compute low-rank matrix $X_1 = U\Sigma V^T$ and sparse error matrix E_1 by $X^{a'} = X^a \odot \hat{\mathbf{M}}$.
Set $w_j^X = \frac{1}{diag(\Sigma)_j + \epsilon_X}$, $w_i^e j = \frac{1}{|E_{1,ij}| + \epsilon_E}, \epsilon_E = 10^{-3}$.
while $\frac{\|X_{k+1,(i,j)\in\Omega} - X_{k,(i,j)\in\Omega}\|_F}{mn} > \varepsilon$ and $k < maxiter$ **do**
  Using Algorithm 2 with parameter W^X and W^e to Compute X_{k+1} based on $X^{a'}$.
  set $X^{a'} = X^{a'} \odot \hat{\mathbf{M}} + X_{k+1} \odot (\mathbf{I} - \mathbf{M})$.
  $k = k + 1$.
end while
Output $X = X_k$

Although the above matrix completion problem (3.11) can be simplified as matrix decomposition (3.13), the corresponding elements of $X_{(i,j)}$ for pixels $\mathbf{M}(i,j) = 0$ were different from their original values in $X^a_{(i,j)}$. To address this issue, we use fixed points iterative method to solve our proposed optimization problem. Our proposed iterative algorithm, referred to as Reweighted Low-rank Keypoints Completion (RLKC), is summarized in Algorithm 2. Algorithm 1 summaries this algorithm for solving the inner optimization problem of smoothed and reweighted low-rank matrix recovery, which obtains the smoothed low-rank matrix X by iterative alternating direction method.

3.4.3 Algorithm Convergence Analysis

We recognize that it is challenging to obtain a theoretical analysis of the algorithm convergence since the objective function is non-convex. Instead, we study its convergence experimentally. In our proposed iterative alternating solution, each variable is updated within one iteration, and the updated variables will be used for the next iterations. To study its convergence, we monitor the relative error $\mathcal{E}_r$

$$\mathcal{E}_r = \frac{\|X^a - Z - E\|_F + \|X - Z\|_F}{\|X^a\|_F} \tag{3.25}$$

In Fig.3.4, we plot this relative error as the number of iteration increases for four sample images. We can see that our algorithm converges very fast after few iterations.

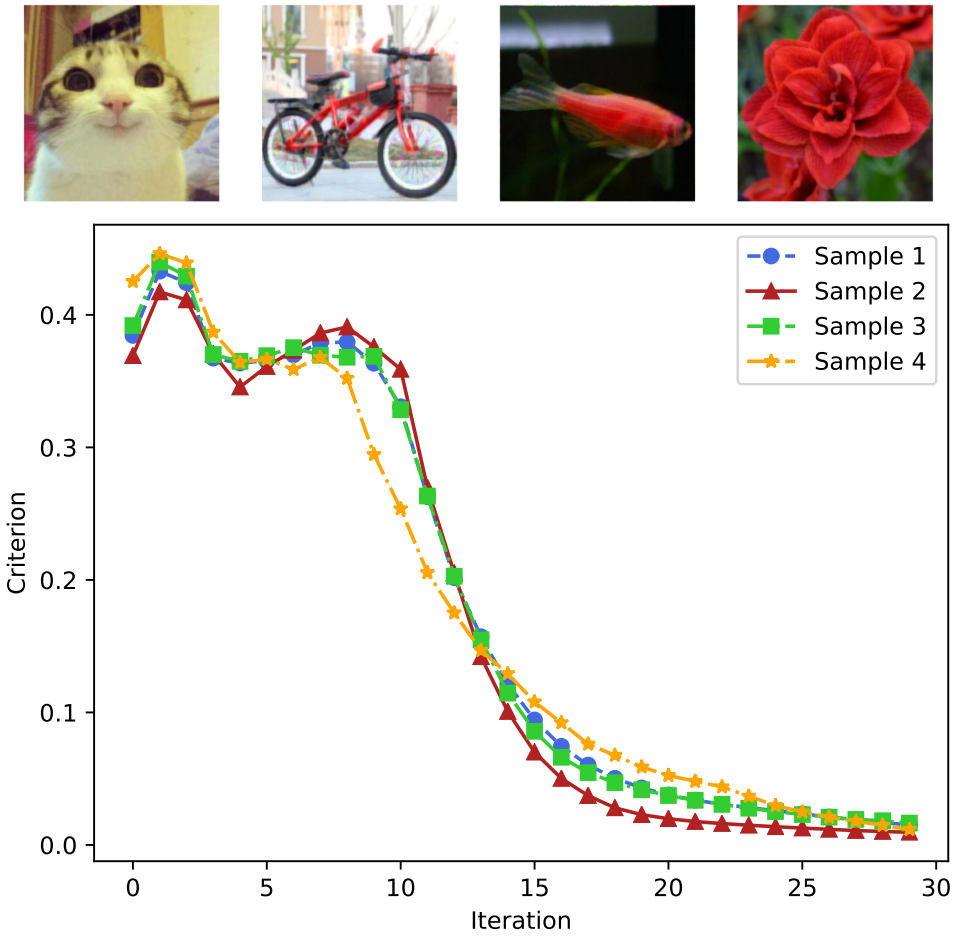


Figure 3.4: Convergence analysis of our iterative alternating algorithm over 4 masked images.

3.5 Experimental Results

In this section, we present experimental results to evaluate our defense method based on low-rank completion of high-sensitivity points and compare its performance with state-of-the-art methods.

3.5.1 Experimental Setup

In this work, we compare our defense method against two state-of-the-art network defense methods based on image noise removal or pre-processing: *total variance minimization* [24] and *image quilting* [24]. We choose these two because, similar to our paper, they all fall into the category of image denoising, aiming to remove the adversarial attack noises and recover the original image. Unlike many other defense methods [72, 68, 26], they do not introduce any modification to the target network or add additional network to the target network. In order to provide a fair comparisons, we do not take the results from [24] directly, but reproduce their defense instead via the code released by [28] in the comparison. This is because the baseline models in [24] are not provided, and the difference of the baseline models will result in different adversarial images and defense performance. In this work, we refer to the total variance minimization and image quilting as *TVM* and *QUILT* algorithms, respectively.

We evaluate and compare all defenses on the CIFAR-10 and SVHN (Street View House Number) datasets [62, 90]. The CIFAR-10 dataset consists of 60,000 natural images in 10 classes, with 32×32 image size and the Street View House Numbers (SVHN) dataset consists of about 200K street numbers images.

We evaluate these defenses against gray-box attacks, black-box attacks, and white-box attacks. In our gray-box and black-box experiments, two fast-gradient attack methods and one optimal attack method are selected: FGSM, BIM, and DeepFool. We use the $\mathcal{L}_\infty$ norm to measure the strength of adversarial perturbations in FGSM and BIM, the maximum perturbation range is denoted as $\epsilon = 8$. Following other papers, we set the maximum iteration of BIM attacks to be 10. In DeepFool, the maximum iteration is set as 5 and the overshoot is set as 0.02. In white-box attacks, we apply the BPDA and normalized $\mathcal{L}_2$ loss (referred to as NL2) attack algorithm to evaluate all defenses. In BPDA, ϵ is set as 2 and maximum number is set as 10.

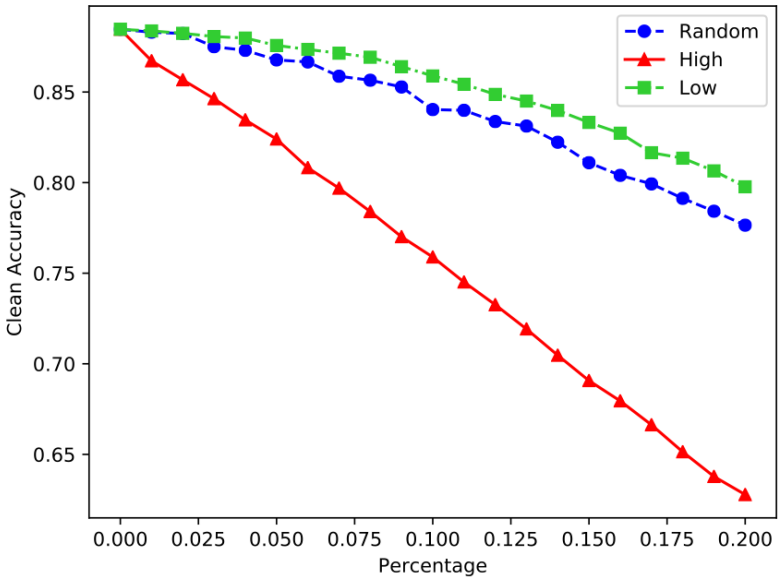


Figure 3.5: Accuracy of removed selected points on clean CIFAR-10 images.

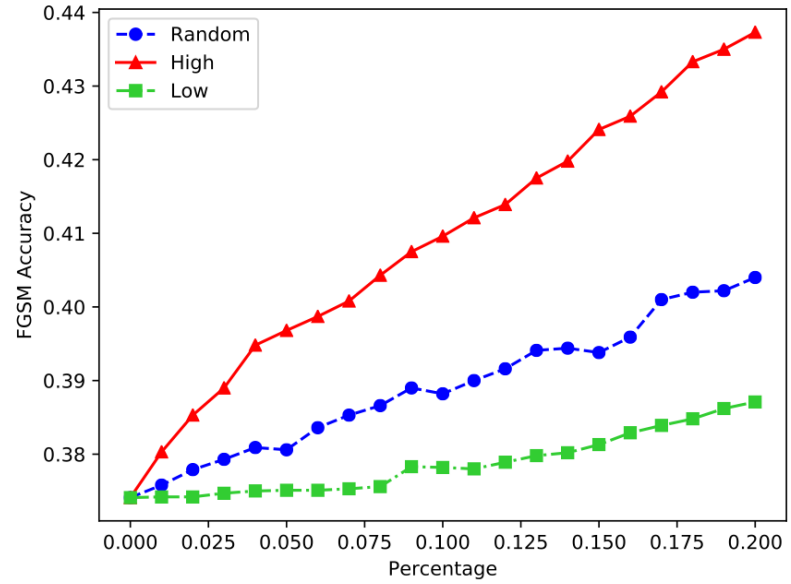
We implement all attacks and defenses using TensorFlow [91]. FGSM, BIM and DeepFool images are generated by the CleverHans [92] and we utilize the code from [28] to generate BPDA-attacked images. All the algorithms run on a desktop computer with an Intel core i7-7800X 3.50 GHz CPU, one Nvidia GTX 1080 Ti GPU, 64

GB of RAM, and Ubuntu 18.04.

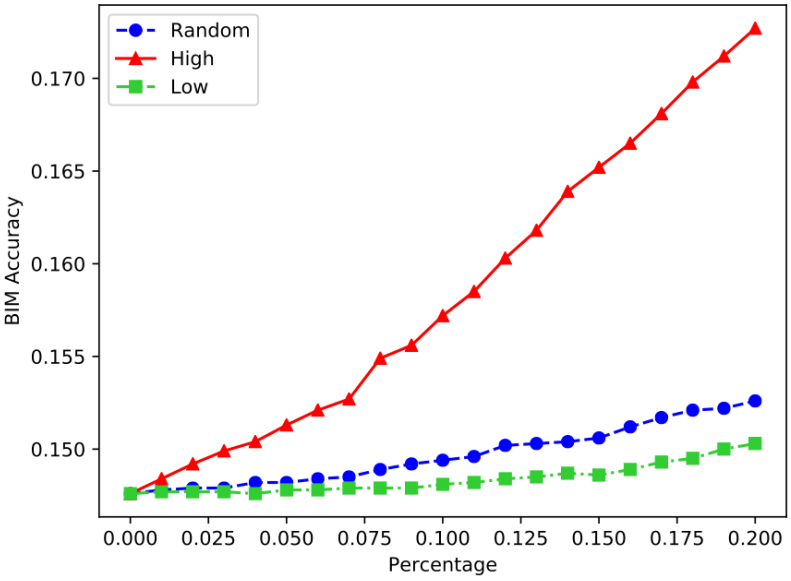
3.5.2 Evaluating the Performance of High-Sensitivity Keypoints Detection Algorithm

In the following experiments, we aim to further understand the behavior and performance of our high-sensitivit keypoint detection algorithm. Our experiments are performed on the CIFAR-10 dataset. A pre-trained VGG16 model with is used as the baseline model whose classification accuracy is 88.46%. We compare three different keypoints selection methods: (a) *random selection*, (b) *selecting the high-sensitivity keypoints*, and (c) *selecting the low-sensitivity keypoints*. Once the keypoints are selected, they are removed from the original image and repaired with Gaussian smoothing. We then pass these repaired images into the classification network. In Fig. 3.5, we plot the classification accuracy of the result image with respect to the percentage of removed keypoints using the above three keypoint selection methods. We can see that, by removing the high-sensitivity keypoints, the image classification accuracy has the largest drop when compared to the other two methods. This indicates that the selected keypoints have larger contribution to the image classification performance than other image pixels.

Next, we study how the keypoint selection methods affect the classification accuracy of attacked images. During the experiments, using the above three methods, we select high-sensitivty keypoints from the attacked images and replace them using te corresponding pixels from their original images. In this way, we have removed the adversarial attack noise at these selected keypoints. We then pass the result images into the classification network. Fig. 3.6 shows the classification accuracy results with



(a) Accuracy of FGSM images



(b) Accuracy of BIM images

Figure 3.6: Accuracy of removed selected points on signed gradient-based adversarial CIFAR-10 images.

two gray-box attack methods, FGSM (top) and BIM (bottom). We can see that, if we remove the adversarial attack noise at locations selected by the high-sensitivity keypoints method, the classification accuracy is much higher than the other two methods. This indicates that our method is able to identify those image pixels which have the most contribution to the classification accuracy. If these pixels are corrupted by the adversarial attack, the image classification accuracy has the largest drop.

3.5.3 Evaluating the Defense Performance on the CIFAR-10 Dataset

We train two network models, VGG-16 [43] and ResNet-18 [4] for image classification on the CIFAR-10 dataset which achieve classification accuracy of 88.46% and 87.06%, respectively. In the gray-box and black-box attack and defense experiments, we select the top 5% of image pixels as high-sensitivity keypoints. The hyper-parameters λ and η are set as 8 and 0.2 respectively.

Defense Against Gray-Box Attacks

The defense results against gray-box attacks are summarized in Table 3.1. We compare our method against two image restoration-based defense methods: TVM and QUILT. We can see that our algorithm is competitive with these two state-of-the-art methods. For the BIM attack method, our algorithm performs the best. TVM introduces the least damage to the original clean image and can provide good protection for those adversarial images with small perturbation, such as DeepFool. But its performance degrades significantly with large attack noise. QUILT and our algorithm can defend large attacks more effectively. Fig. 3.7 shows some example images from

Methods	No Defense	TVM [24]	QUILT [24]	Ours
Clean	87.06	**82.71**	77.49	77.60
FGSM	14.18	25.51	32.43	**34.25**
BIM	7.56	16.18	25.33	**28.03**
DeepFool	8.64	**76.60**	71.60	71.00

Table 3.1: Gray-box Defenses on CIFAR-10

the CIFAR-10 dataset and the subjective performance of their defense results with these three defense methods. The first row shows the attacked image by the BIM method. The second, third, and fourth rows show the restored images by the TVM, QUILT, and our defense methods. The last row shows the original clean images. We can see that our algorithm is able to successfully remove the attack noise, maximally restore the visual quality, and match the original images best. In the result images obtained by TVM and QUILT, we can still clearly see artificial image noise.

Figure 3.7: CIFAR-10 Samples: the first raw is the BIM images, the second row is the TVM images, the third row is QUILT images, the fourth raw is our keypoints completed images and the last row is the clean images.

Defense Against Black-Box Attacks

The black-box attack and defense results are summarized in Table 3.2. Our method outperforms the other two methods for both FGSM and BIM attacks. Our high-sensitivity low-rank image completion defense shows the much stronger generalization ability than other the twos in black-box attack experiment because the high-sensitivity points are obtained by the test model instead of the attack model.

Methods	No Defense	TVM [24]	QUILT [24]	Ours
Clean	87.06	**82.71**	77.49	77.60
FGSM	17.75	52.33	50.56	**57.77**
BIM	7.54	42.57	50.11	**55.40**

Table 3.2: Black-box defenses on CIFAR-10

Defense Against White-Box Attacks

We evaluate the performance of our defense method with white-box attacks on the CIFAR-10 dataset using the ResNet-18. We choose the BPDA attack method which is one of the most powerful white-box attack methods developed in the literature [28]. Table 3.3 summarizes the defense results under the BPDA attacks with and without the NL2 loss function. We can see that our algorithms significantly outperforms the TVM and QUILT defense methods by a large margin of more than 24%.

Table 3.3: White-box defenses on CIFAR-10

Methods	TVM [24]	QUILT [24]	Ours
BPDA [28]	33.67	36.31	**61.60**
BPDA with NL2[28]	34.08	36.91	**61.63**

3.5.4 Evaluating Defense Performance on the SVHN Dataset

In this experiment, we evaluate the performance of our algorithm on the SVHN dataset. The baseline accuracy of the VGG-16 [43] and ResNet18 [4] models on this dataset are 92.63% and 93.82%, respectively. We select top 10% pixels as the high-sensitivity points in all experiments on SVHN.

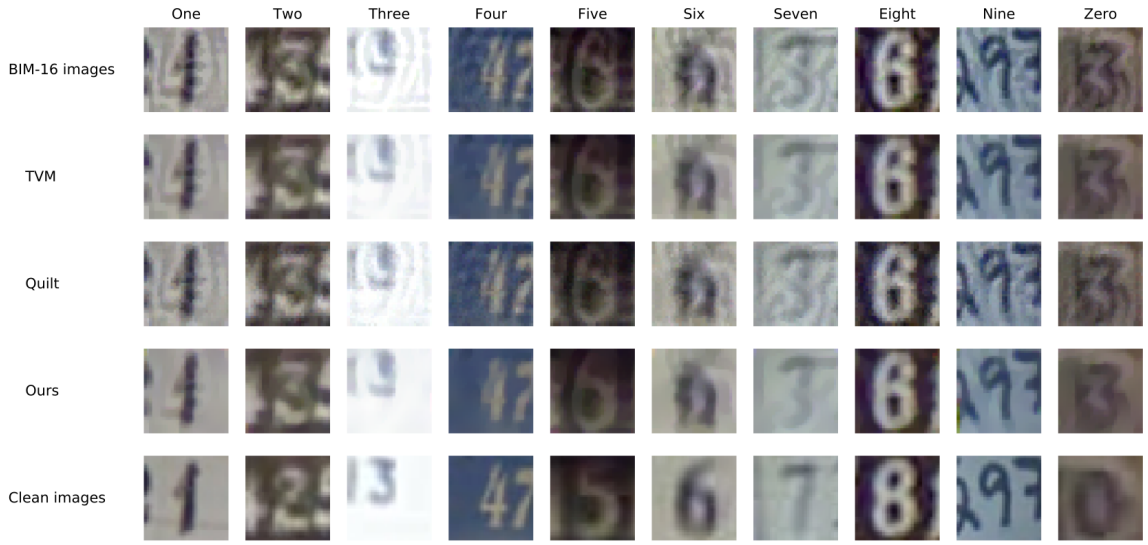


Figure 3.8: SVHN Samples: the first raw is the BIM images, the second raw is the TVM images, the third raw is QUILT images, the fourth raw is our keypoints completed images and the last raw is the clean images.

Table 3.4 summarizes the the defense results with the ResNet-18 model. On this dataset, the QUILT method has the least damage to the original clean image. Our algorithm drops the classification accuracy by about 2%. For all attack methods, our algorithm consistently outperforms the other two methods by large margins.

Fig. 3.8 shows some image examples and their defense results from the SVHN dataset. The first row shows the images being attacked by the BIM method. Rows 2, 3, and 4 show the result images obtained by the TVM, QUILT, and our defense

methods. The last row shows the clean images. We can see that our algorithm can effectively remove the adversarial noise and restore the image visual quality, and match the original images best.

Methods	No Defense	TVM [24]	QUILT [24]	Ours
Clean	93.82	92.59	**93.30**	91.52
FGSM	18.03	27.09	21.73	**35.75**
BIM	10.32	20.35	13.84	**31.11**
DeepFool	3.89	70.77	53.37	**76.48**

Table 3.4: Gray-box Defenses on SVHN

We use the same black-box defense setting as those experiments on the CIFAR-10 dataset, *i.e.* ResNet18 model as the attack model and VGG16 as the test model. Defense results is shown in Table 3.5. The QUILT algorithm still performs the best on clean images. But, on the attacked images, our method significantly outperforms the other two methods by a large margin. For example, for the FGSM attack, our method has improved the accuracy by about 10% over the TVM method and 20% over the QUILT method.

Table 3.6 summarizes the defense results under the white-box BPDA attacks with and without NL-2 loss. Our defense method obtains 72.00% accuracy against BPDA attack only and 72.01% accuracy against BPDA with NL2 attack, which shows a significant improvement as it does on CIFAR-10. We can see that our proposed method achieves the most performance gain over strong attacks, such as the BPDA white-box attackes.

Table 3.5: Black-box defenses on SVHN

Methods	No Defense	TVM [24]	QUILT [24]	Ours
Clean	93.82	92.59	**93.30**	91.56
FGSM	38.93	48.17	42.47	**54.48**
BIM	31.36	46.55	36.99	**56.80**

Table 3.6: White-box defenses on SVHN

Methods	TVM [24]	QUILT [24]	Ours
BPDA [28]	32.02	29.90	**72.00**
BPDA with NL2[28]	31.84	29.46	**72.01**

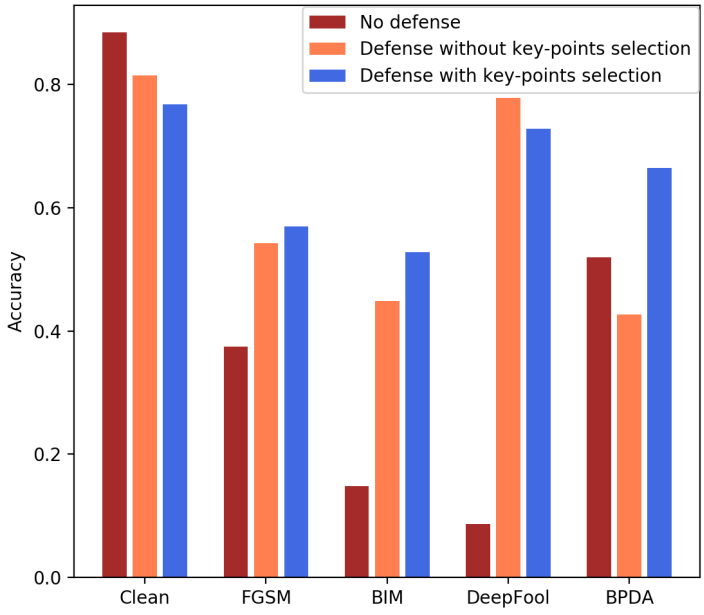


Figure 3.9: Ablation study on CIFAR-10.

3.5.5 Ablation Studies

In the following ablation study, we conduct experiments to analyze the contribution of major algorithm components. Specifically, we evaluate the following three methods: (1) no defense, (2) TV-norm based image smoothing without high-sensitivity keypoint selection and (3) with keypoint selection. We perform experiments on the CIFAR-10 dataset. We use VGG-16 as the attack and test model and set the magnitude of the FGSM and BIM perturbation as $\epsilon = 8$. Fig. 3.9 shows the results of the defense with or without our high-sensitivity points. Although the accuracy of clean images and DeepFool images is a little bit lower, our completion indeed improve the most defenses performances, especially on defense against the BPDA.

Chapter 4

Structure-Preserving Progressive Low-Rank Image Completion for Defending Adversarial Attacks

4.1 Motivation

Deep neural networks map the input image pixels into a decision output to classify images, recognize objects, and achieve many other vision analysis tasks. Based on local filtering and pooling, it analyzes pixel values and texture details in each image neighborhood, gradually summarizes the information over the network layers, and produces the final decision at the output layer. Recently, researchers have recognized that deep neural networks are often bias towards image textures instead of semantic structures and global visual cues [1]. For example, Figure 4.1 (a) shows an image of dog and (b) shows a texture patch of an Indian elephant. (c) is synthesized from (a) and (b). Deep neural networks, for example, those pre-trained on ImageNet, will

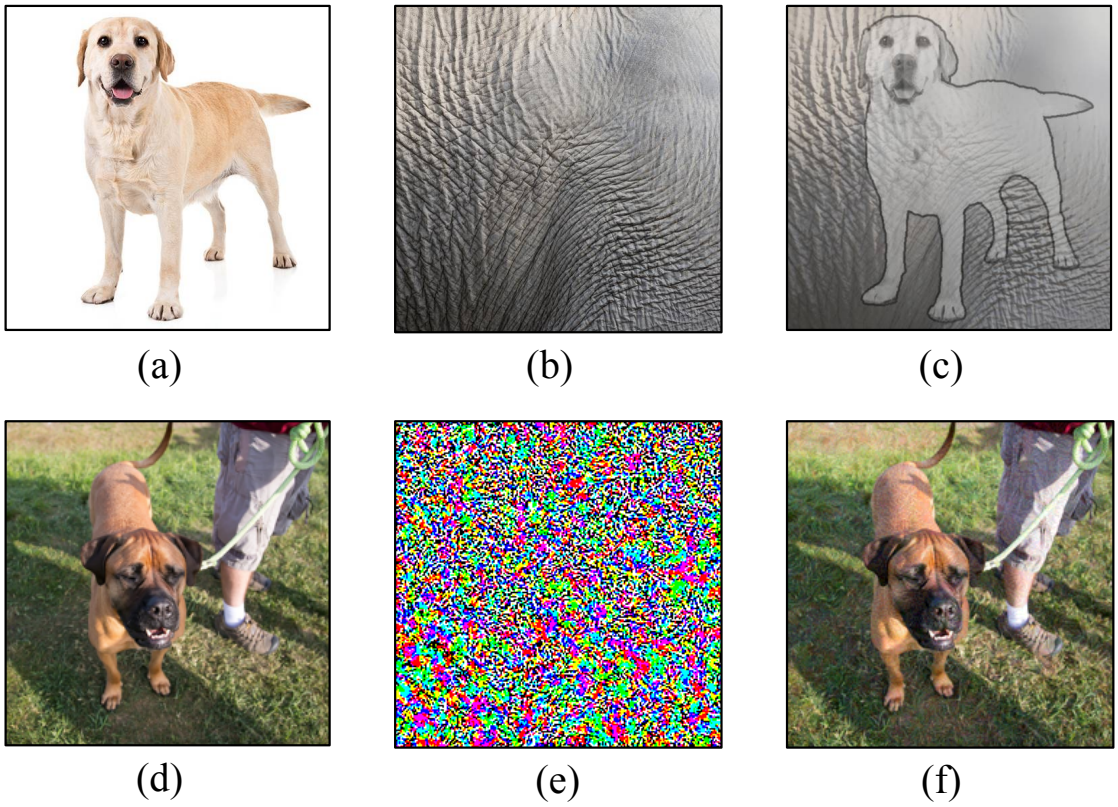


Figure 4.1: (a) a dog image; (b) an India Elephant skin image; (c) the synthesized image of dog from (a) and (b) which is classified as an Indian Elephant [1]; (d) the original dog image; (e) the adversarial attack noise generated by the FGSM white-box attack [2]; and (f) the attacked image being classified as goose.

often mis-classify image (c) as an Indian elephant. However, human eyes can easily tell that it is still a dog. This suggests that deep neural networks often build their final decision largely upon local image textures, instead of the global object structures, for example, shapes.

Figure 4.1 shows another classic example in adversarial attacks [2]: (d) is the original dog image and (e) is the adversarial noise pattern generated by the attack method [2]. The maximum change to each pixel is controlled under 0.7% of the pixel value range. If we add this very small noise onto the original dog image, the

deep neural network will classify the result image (f) as goose which is a totally different animal. However, our human eyes have no problem at all in recognizing (f) as a dog since the overall semantic structures are the same. This is because, the adversarial noise is uniquely designed so that the error will accumulate along the network inference path, exceed the final decision threshold, and produce a wrong output.

This example suggests that there is a significant semantic gap between deep neural networks and human visual systems; the emphasis on and bias towards local image texture details cause the network to be prone to adversarial attacks. In the meantime, it also suggests a very interesting approach for defending deep neural networks against adversarial attacks: making the network focus more on semantic structures and global visual cues, instead of local details since the adversarial attacks operate on local pixels and modify their detailed values.

To implement this idea, one possible approach is to develop a structure-preserving image smoothing method to pre-process the image. At the training side, these images are added to the training set to improve the network's capability in capturing global structures. At the test side, this structure-preserving image smoothing will largely remove the adversarial noise hidden in local image textures. Coupled with the structure-oriented training, this method will be able to successfully defend deep neural networks against adversarial attacks.

In this work, we propose to develop a structure-preserving progressive low-rank image completion (SPLIC) method to remove unneeded texture details from the input images and let the deep neural network focus more on global object structures and semantic cues. We formulate the problem into a low-rank matrix completion

problem with progressively smoothed rank functions to avoid local minimums during the optimization process. We include total variation constraint to further enhance the capability of our method to capture object structures. Our experimental results demonstrate that the proposed method is able to successfully remove the insignificant local image details and let the network learning focus on global object structures during the learning process. On black-box, gray-box, and white-box attacks, our method outperforms existing defense methods and significantly improve the adversarial robustness of the network.

4.2 Related Work

Adversarial examples has already been described in Chapter 3: fast-gradient based attack methods (*i.e.* *FGSM*, *BIM* and *PGD*), optimal attack methods (*i.e.* *DeepFool* and *CW*) and the state-of-the-art white-box attack *BPDA* [28]. We classify the defending method into two categories: image de-noising based defenses (*i.e.* image transforms in [93], PixelDefend [25] and Defense-GAN[26]), and adversarial training methods (*i.e.* [94, 69]).

Recently, some new adversarial attacks and many new defending works are proposed to overcome the vulnerability of deep neural networks. Our new proposed method belongs to image pre-processing as well.

4.2.1 Recent adversarial attacks

A finding that convolutional networks are sensitive to the directions of Fourier basis functions is revealed in [95]. They develop a new adversarial attack method to create

shift-invariant universal adversarial perturbations available in black-box condition. A interesting application on face recognition [96] is proposed in black-box setting, which can model the local geometries of the search directions and reduce the dimension of the search space. Beside the transfer-based black-box attacks by the substitute model, boundary attack [97] is proposed, which seeks to reduce the perturbation from a relatively large adversarial. This attack is conceptually simple and close to no hyperparameter tuning. Its black-box attack doesn't need the substitute model and is competitive with the best gradient-based attacks in standard computer vision tasks. A state-of-the-art SPSA attack [98] experimentally validate the "security by obscurity" nature of recently proposed defense methods and dramatically reduce the performance of these defenses.

4.2.2 Recent de-noising methods

Sun *et al.* [99] firstly exploits convolutional sparse coding and develops the *Sparse Transformation Layer (STL)* to project images to the quasi-natural image space. Ensemble image transforms strategy is applied to defend against adversarial attacks in [100, 101]. A multi-channel randomization is exploited in [100]. Each channel introduces its own randomization in a special transform domain, including DCT transform, and a secret key is shared between the training and test. For the defender side, the shared key keeps the gradients in key-defined sub-spaces, but for the attacker side it stops gradient back propagation and the creation of various bypass systems. Another advantage is the reliability increasing by the aggregation that fuses soft-outputs from all channels. Yang *et al.* [29] proposes a viewpoint that human categorize images by the main structure while the perturbation can not change the main structure of

images. Then they develops *ME-Net* [29] by low-rank approximation to train a model with main image structure. However, *ME-Net* ignores the finer details and will cause inaccurate results. Our method improves the low-rank based images reconstruction and achieve a better adversarial defense. ShieldNets [102] exploits probabilistic adversarial robustness (PAR) to neutralize adversarial attacks by concentrating sample probability to adversarial-free zones. D3 [103] algorithm divide the input image into multiple patches, de-noising each patch independently, and then reconstructing the image. It is also a non-differentiable defense mechanism which makes it non-trivial for gradient-based attacks. Moreover, it does not use adversarial examples to fine-tune the network.

4.2.3 Recent Adversarial Training

For adversarial training, *Feature Scattering* [104] uses an unsupervised adversarial training method to scatter features in the latent space to improve the robustness of deep neural networks. *Metric learning* is also introduced in adversarial training by [104]. It utilizes triplet loss to increase the distance between anchor adversarial samples and positive clean samples but reduce the distance between anchor samples and negative clean samples. Hu *et al.* [105] present a novel viewpoint, which regards the omnipresence of adversarial perturbations as a strength rather than a weakness. Their work shows that if an image has been perturbed, these adversarial directions either become harder to find with gradient methods or have substantially higher density than for natural images. It utilize this property to achieve a much better defense against white-box attack. An fast adversarial training algorithm [106] is presented to eliminate the overhead cost of generating adversarial examples. It recycles the

gradient information computed when updating model parameters and is much faster than other adversarial training methods. It is believed that it is hard to improve networks adversarial robustness unless making networks larger. Xie *et al.* [107] provide evidence to challenge these common beliefs by a careful study about adversarial training. They show that widely-used ReLU activation function's non-smooth nature significantly weakens adversarial training. In their work a smooth adversarial training (SAT) method is proposed by replacing ReLU with its smooth approximations to enhance network robustness. SAT allows networks training procedure to find harder adversarial examples and compute better gradient updates.

4.2.4 Matrix Completion by Smoothed Rank Function

Matrix completion try to reconstruct an unknown large matrix and only a small subset of entries can be observed. Usually, recovery of a matrix is possible if the matrix is low-rank and the number of known entries are sufficient. In this case, the matrix completion problem is to set a matrix with minimum rank to the given entries. Instead of rank function, nuclear-norm function [108, 109] are used due to the non-convexity of rank function. Recently, matrix completion based on smoothed rank function (SRF) is presented in [110, 111]. There are two advantages in comparison to the methods based on nuclear norm: 1. SRF algorithms are faster if the rank of the matrix is known and high; 2. SRF algorithms achieve more accurate results in almost all situations. Generally, SRF methods lie much closer than nuclear norm to rank function due to the smoothing term and is much faster then the nuclear norm. But these iterative algorithms need to decompose the matrix by singular value decomposition (SVD) in each iteration. Furthermore, if the number of unknown entries in the matrix are too

large, these algorithms could not get a good performance. Wang *et al.* [112] propose a faster SRF algorithm to improve the running time significantly and outperform most of matrix completion methods.

4.3 Methods

As illustrated in Figure 4.2, we formulate this problem as a structure-preserving low-rank image completion problem. For each input image $\boldsymbol{X}$ in both training and test sets, we generate a random mask $\boldsymbol{M}$ to select half of the pixels as anchor pixels and the rest as target pixels. We then develop a structure-preserving low-rank image completion (SPLIC) method with total variation constraint to complete the image at target pixel locations using the anchor pixels as constraints. To avoid being trapped into local minimums during rank minimization, we use the method of smoothed rank functions with progressive scale control δ which can adapt to local object structure at different scales. After the target pixels are completed or smoothed with adversarial noise being largely removed, we alternate the anchor and target pixels, and then apply the above SPLIC method remove the adversarial noise at anchor pixels. This SPLIC pre-processing step is applied to all training images and each test image. We expect that the learned deep neural network will focus more on semantic structures instead of local texture details, improving its robustness to adversarial attack noise. In the following section, we explain our proposed SPLIC method in more detail.

4.3.1 Low-Rank Image Completion Based on Nuclear Norm Minimization

Semantic structures of objects and images are inherently low rank [86]. Recently, methods for low-rank matrix approximation have been developed to characterize the low-rank structures in images [112]. In this paper, we propose to formulate the problem of removing adversarial noise from attacked images while preserving important semantic structure information for successful recognition as a low-rank matrix completion problem. Specifically, let $\boldsymbol{X} = [x_{ij}]_{m\times n} \in \mathbb{R}^{m\times n}$ be the original image of size $m \times n$. $\boldsymbol{M} = [m_{ij}]_{m\times n}$ is the random binary mask. If $m_{ij} = 1$, the corresponding image pixel x_{ij} is chosen as the anchor pixel. Otherwise, it is considered as a target pixel. We denoted the set of anchor pixels by Ω. During low-rank image completion, we attempt to estimate and revise the values of target pixels with the fixed anchor pixels as constraints so that the rank of the recovered image $\hat{\boldsymbol{X}} = [\hat{x}_{ij}]_{m\times n}$ is minimized. Specifically, the problem is formulated as

$$\begin{aligned} &\min \text{ rank}(\hat{\boldsymbol{X}}), \\ &s.t.\ x_{ij} = \hat{x}_{ij},\ (i,j) \in \Omega. \end{aligned} \tag{4.1}$$

It should be noted that the rank as a function of the matrix is a highly nonlinear and non-convex function [86], which poses significant challenges for obtaining efficient solutions for the problem in (4.1). More importantly, the solution is often trapped into local minimums. To address this issue, the nuclear norm $||\hat{\boldsymbol{X}}||_*$ of matrix $\boldsymbol{X}$ is often used to approximate the rank of matrices, which leads to a convex minimization problem with highly efficient solutions available. Let $\{\sigma_k(\hat{\boldsymbol{X}})\}$, $1 \le k \le l$, $l = \min(m,n)$, be the set of singular values of matrix $\hat{\boldsymbol{X}}$. Then, the rank of $\hat{\boldsymbol{X}}$ is the

number of non-zero entries in $\{\sigma_k(\hat{\boldsymbol{X}})\}$. However, the nuclear norm is the summation of all singular values. The nuclear norm approximation of the optimization problem in (4.1) is given by

$$\begin{aligned} &\min ||\hat{\boldsymbol{X}}||_* = \sum_{k=1}^{K} \sigma_k(\hat{\boldsymbol{X}}), \\ &s.t.\ x_{ij} = \hat{x}_{ij},\ (i,j) \in \Omega. \end{aligned} \tag{4.2}$$

4.3.2 Progressive Smoothed Rank Functions with Total Variation Constraint

In this work, we have found that the nuclear norm does not provide an effective approximation of the original rank function, especially for images or objects with complex semantic structures at different spatial scales. We observe that, geometrically, smooth terms generally lie much closer to the essential rank function than nuclear norm. In the meantime, we wish to take advantage of the convex nature of the nuclear norm so that the optimization process will not be trapped into local minimums. To address this issue, we propose to use the smoothed rank function method developed in [110] to better preserve the important structure information. Given a matrix $\hat{\boldsymbol{X}}$, its smoothed rank function is defined based on Gaussian smoothing of its singular values:

$$F_\delta(\hat{\boldsymbol{X}}) = l - \sum_{k=1}^{l} e^{-\frac{\sigma_k^2(\hat{\boldsymbol{X}})}{2\delta^2}} \tag{4.3}$$

This smoothed rank function well approximates the original rank of the matrix, when δ approaches 0. For example, considering a matrix $\hat{\boldsymbol{X}}$ with a rank of k_0. The first k_0 singular values are positive, $\sigma_k(\hat{\boldsymbol{X}}) > 0$ for $k \leq k_0$. The rest singular values are zeros,

$\sigma_k(\hat{\boldsymbol{X}}) = 0$ for $k_0 < k \leq l$. In this case, when $\delta \to 0$, we have

$$e^{-\frac{\sigma_k^2(\hat{\boldsymbol{X}})}{2\delta^2}} = \begin{cases} 0, & 0 \leq k \leq k_0, \\ 1, & k_0 < k \leq l, \end{cases} \tag{4.4}$$

Therefore, according to (4.3), $F_\delta(\hat{\boldsymbol{X}}) = k_0$. Figure 4.2 (right) shows the smoothed rank function with progressive control δ. When δ is large, it is a convex function. Based on this rank function, the algorithm can guide the optimization towards the region of global minimum. Then, the method gradually reduces the value of δ and refines the scale of gradient search. This progressive optimization can successfully avoid the local minimum while enjoying the advantage of local convex optimization. This gradual tuning technique for minimizing non-convex functions is referred to as graduated non-convexity [110].

In this work, we observe that the smoothed rank function can obtain better performance in matrix completion than nuclear norm which was used in the ME-Net method [29]. However, it still suffers from performance degradation when the image has high intrinsic rank structures or has noise density. It is not able to efficiently remove sparse adversarial noise with high density due to the absence of an proper regularization scheme. Furthermore, they cannot effectively maintain the smoothness of local neighborhood pixels in the smooth regions affected by adversarial noise. To address this issue, we propose to incorporate the total variation (TV) constraint [113] into the progressive smoothed rank optimization problem. Mathematically, our

SPLIC optimization problem can be formulated as:

$$\begin{aligned} &\min_{\hat{\boldsymbol{X}}} F_\delta(\hat{\boldsymbol{X}}) + \lambda \cdot C(\hat{\boldsymbol{X}}) \\ &s.t.\ x_{ij} = \hat{x}_{ij},\ (i,j) \in \Omega. \end{aligned} \tag{4.5}$$

where λ is a weighting parameter which will be analyzed in our ablation studies. $C(\hat{\boldsymbol{X}})$ is the TV constraint. Since the original TV-norm is hard to compute the gradient directly, we rewrite the TV constraint function as follows:

$$\begin{aligned} C(\hat{\boldsymbol{X}}) = &\sum_{i=1}^{m-1}\sum_{j=1}^{n-1} \frac{(\hat{x}_{i,j}-\hat{x}_{i+1,j})^2+(\hat{x}_{i,j}-\hat{x}_{i,j+1})^2}{2} \\ &+\sum_{i=1}^{m-1}\frac{(\hat{x}_{i,n}-\hat{x}_{i+1,n})^2}{2}+\sum_{j=1}^{n-1}\frac{(\hat{x}_{m,j}-\hat{x}_{m,j+1})^2}{2}, \end{aligned} \tag{4.6}$$

where the first entry in the summation are variations for pixels inside the image and the last two entries are for pixels on the horizontal and vertical edges.

4.3.3 Solution to the SPLIC Optimization Problem

In this section, we derive a gradient descent-based numerical solution for the SPLIC optimization problem in (4.5). To this end, we need to determine the derivatives of $F_\delta(\hat{\boldsymbol{X}})$ and $C(\hat{\boldsymbol{X}})$ with respect to $\hat{\boldsymbol{X}}$.

We first introduce the definition of ***absolutely symmetric function*** [114]. Given a vector $\boldsymbol{\gamma}$ in $\mathbb{R}^q$, we sort its vector elements in a non-increasing order to form a new vector $\hat{\boldsymbol{\gamma}}$. A function $f : \mathbb{R}^q \to \mathbb{R}$ is absolutely symmetric if $f(\gamma) = f(\hat{\gamma})$

for any vector γ in $\mathbb{R}^q$. For matrix $\hat{\boldsymbol{X}}$, its singular value decomposition (SVD) is

$$\hat{\boldsymbol{X}} = \boldsymbol{U} \cdot \text{diag}\{\sigma_1(\hat{\boldsymbol{X}}), \cdots, \sigma_l(\hat{\boldsymbol{X}})\} \cdot \boldsymbol{V}^T, \tag{4.7}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are right and left singular vector matrices. We can see that the following function

$$F_\delta(\boldsymbol{z}) = l - \sum_{k=1}^{l} e^{-\frac{z_l^2}{2\delta^2}} \tag{4.8}$$

is absolutely symmetric. According to the Theorem 3.1 of [114] and [110], the sub-gradient of $F_\delta(\hat{\boldsymbol{X}})$ can be calculated as follows:

$$\nabla F_\delta(\hat{\boldsymbol{X}}) = \boldsymbol{U} \cdot \text{diag}\{\frac{\sigma_1}{\delta^2} e^{-\frac{\sigma_1^2}{2\delta^2}}, \cdots, \frac{\sigma_l}{\delta^2} e^{-\frac{\sigma_l^2}{2\delta^2}}\} \cdot \boldsymbol{V}^T. \tag{4.9}$$

For the total variation term $C(\hat{\boldsymbol{X}})$ in equation (4.6), its derivative with respect to $\hat{\boldsymbol{X}}$ is given by

$$\nabla C(\hat{\boldsymbol{X}}) = \begin{cases} 2\hat{x}_{i,j} - \hat{x}_{i+1,j} - \hat{x}_{i,j+1}, & \text{inside pixels} \\ \hat{x}_{i,j} - \hat{x}_{i,j+1}, & i = m \\ \hat{x}_{i,j} - \hat{x}_{i+1,j}, & j = n \end{cases} \tag{4.10}$$

With the gradient of $F_\delta(\hat{\boldsymbol{X}})$ and the gradient of $C(\hat{\boldsymbol{X}})$ being obtained by equations (4.9) and (4.10), we are ready to use gradient descent algorithm to solve the problem (4.5).

4.3.4 Alternated SPLIC and Algorithm Summary

In our SPLIC method, we randomly select 50% of pixels as anchor points. At these anchor points, the pixel values are fixed as constraints in the optimization problem (4.5). We observe that these constraints are very important for the robustness of our SPLIC method to avoid algorithm divergence. Once the rest 50% pixels have been re-estimated by our SPLIC method, we will alternate the SPLIC process, using them as the anchor points, and re-estimate the values of the original anchor points. The SPLIC algorithm is summarized in Algorithm 4.

Algorithm 4 structure-preserving progressive low rank image completion (SPLIC)

Require: $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, $\boldsymbol{M} \in \mathbb{R}^{m\times n}$, pre-defined rank $r \leq l$, criteria ε and the maximum iteration number maxiter.

1: **Initialize:** $\hat{\boldsymbol{X}}_t = \boldsymbol{X} \odot \boldsymbol{M}$, $\lambda = 0.02$, $\rho = 0.45$, $\mu = 0.5$, $t = 0$, and δ is set as the largest singular value of $\hat{\boldsymbol{X}}_t$.
2: **while** $\frac{\|\hat{\boldsymbol{X}}_{t+1} - \hat{\boldsymbol{X}}_t\|_F}{mn} > \varepsilon$ and $t <$ maxiter **do**
3: **for** $i = 1$ to 7 **do**
4: Compute SVD of $\hat{\boldsymbol{X}}_t$;
 $\hat{\boldsymbol{X}}_t = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T, \boldsymbol{S} = \text{diag}\{\sigma_1, \cdots, \sigma_l\}$;
5: Set $\sigma_{r+1}, \cdots, \sigma_l$ to zeros,
 $\boldsymbol{S} = \text{diag}\{\sigma_1, \cdots, \sigma_r, 0, \cdots, 0\}$;
6: Compute the gradient of $F_\delta(\hat{\boldsymbol{X}}_t)$ by equation (4.9);
7: Compute the gradient of $C(\hat{\boldsymbol{X}})$ by equation (4.10);
8: Update $\widetilde{\boldsymbol{X}}_{t+1}$:
 $\widetilde{\boldsymbol{X}}_{t+1} = \hat{\boldsymbol{X}}_t - \mu(\nabla F_\delta(\hat{\boldsymbol{X}}_t) + \lambda \nabla C(\hat{\boldsymbol{X}}_t))$;
9: Compute the projection:
 $\hat{\boldsymbol{X}}_{t+1} = (\boldsymbol{1} - \boldsymbol{M}) \odot \widetilde{\boldsymbol{X}}_{t+1} + \boldsymbol{M} \odot \boldsymbol{X}$;
10: $t = t + 1$;
11: **end for**
12: Update the smoothness parameter: $\delta = \rho\delta$;
13: **end while**

Ensure: $\hat{\boldsymbol{X}} = \hat{\boldsymbol{X}}$.

4.4 Experimental Results

In this section, we follow the procedures in existing papers to evaluate the performance of our SPLIC method and compare its performance with the state-of-the-art methods.

4.4.1 Experimental Settings

We evaluate all defense performance in the following three attack scenarios: (1) white box attackers where the attacker has full knowledge about the network and the defense method, (2) black-box attackers where the attacker has no knowledge about the network and the defense network, and (3) gray-box attackers where the attacker knows the network but does not know the defense method. In our experiments, we conduct performance comparison with existing papers on four attack methods, the FGSM, PGD and BPDA, and CW [115] methods, as reviewed in the Related Work section. We use the publicly available package FoolBox [116] for implementation of these attackers. Following prior papers, we conduct performance comparisons on two benchmark datasets, the **CIFAR-10** [62] and the **SVHN** datasets [90].

All the algorithms run on a desktop computer with an Intel core i7-7800X 3.50 GHz CPU, one Nvidia GTX 1080 Ti GPU, 64 GB of RAM, and Ubuntu 18.04. The perturbation in all adversarial attacks are constrained within an ϵ-ball based on the L_∞ distance and we set $\epsilon = 8/255$. In iterative attack methods of PGD and BPDA, we set the single step size to $2/255$ and set the number of iterations to 7. In the CW attack, we fix the confidence level $\kappa = 20$ and the binary search step size as 5. The learning rate and the number of iterations are set as 0.005 and 1000 respectively.

4.4.2 Performance Comparison with Existing Methods

In the following, we evaluate our SPLIC methods under three different attack scenarios and compare its performance with existing methods.

(1) Defense against black-box attackers.

Table 4.1 summarizes the defense performance of our SPLIC methods under three different black-box attacks: FGSM, PGD, and CW on the CIFAR-10 and SVHN datasets, and performance comparisons with existing methods. We also include the results on the clean images without any attacks. The Vanilla method means no defense is applied. We can see that on these black-box attacks, our SPLIC method outperforms the current best method, ME-Net [29]. It should be noted that performance gain is not very significant because the defense accuracy is already very high, very close to the accuracy on the clean images without any attack. For example, on the CIFAR-10, the best accuracy on the clean images is 94.9%. Under the powerful CW attack, our SPLIC method can achieve the accuracy of 93.6%. On the SVHN dataset, this gap is even smaller. We can also see that on the clean images, our method is able to maintain the important semantic structure information for recognition, achieving near the best accuracy. Some methods do not report the results of FGSM and CW, thus we represent them by $-$ in the table.

(2) Defense against white-box attacks.

Table 4.2 summarizes the performance of our SPLIC method under the white-box BPDA attack on the CIFAR-10 and SVHN datasets. The attack method knows both the network and the defense methods. We can see that our SPLIC method

Table 4.1: Defense performance under black-box attacks.

CIFAR-10				
Method	Clean	FGSM	PGD	CW
Vanilla	93.4%	24.8%	7.6%	9.3%
Madry [94]	79.4%	67.0%	64.2%	78.7%
Thermometer [75]	87.5%	–	77.7%	–
TLA-RN [117]	81.0%	–	66.0%	–
TLA-SA [117]	86.2%	–	61.7%	–
TLA [117]	86.2%	–	70.6%	–
ME-Net [29]	**94.9%**	**92.2%**	91.8%	**93.6%**
SPLIC-Net (Ours)	94.0%	91.0%	**92.2%**	**93.6%**
SVHN				
Vanilla	95.0%	31.2%	8.6%	20.4%
ME-Net [29]	**96.0%**	91.8%	91.1%	95.5%
SPLIC (This Work)	95.8%	**92.9%**	**93.0%**	**95.7%**

outperforms existing state-of-the-art methods by a large margin. For example, on the CIFAR-10 dataset, it improves the accuracy by 8.7%. On the SVHN dataset, the performance gain is 6.1%, which is quite significant. We can also see that the white-box BPDA attack is very powerful, making it much more challenging to defend. This is because the white-box attack has full knowledge about the defense method and has the opportunity to learn the behavior of the defense method and re-adjust the adversarial attack noise.

(3) Defense against gray-box attacks.

Table 4.3 summarizes the results on the CIFAR-10 and SVHN under gray-box attacks without network training. We compare our method against three defense methods under the gray-box attack scenarios. Since the ME-Net [29] only reported the results after the training on reconstructed images. We used their released code to obtain the results in Table 4.3. We can see that our SPLIC method outperforms existing

Table 4.2: Defense performance under white-box attacks.

Dataset	Method	BPDA
CIFAR-10	Vanilla	0.0%
	TV Mnimization [24]	14.7%
	TLA-RN [117]	52.5%
	TLA-SA [117]	53.5%
	TLA [117]	53.9%
	ME-Net [29]	59.8%
	SPLIC (This Work)	**68.5%**
	Gain	**+8.7%**
SVHN	Vanilla	0.0%
	Madry [94]	52.5%
	ME-Net [29]	74.7%
	SPLIC (This Work)	**80.8%**
	Gain	**+6.1%**

methods by large marings. On the CIFAR-10 and SVHN datasets, the performance gains under the PGD attack are 12.6% and 9.1%, respectively.

Table 4.3: Defense performance under gray-box attacks without network training.

Dataset	Method	FGSM	PGD
CIFAR-10	USVT	12.1%	12.1%
	Soft-Imp	30.1%	27.3%
	ME-Net	31.2%	31.3%
	SPLIC	**40.9%**	**43.6%**
	Gain	**+9.7%**	**+12.6%**
SVHN	USVT	36.7%	34.2%
	Soft-Imp	56.1%	55.6%
	ME-Net	55.4%	54.5%
	SPLIC	**59.8%**	**63.6%**
	Gain	**+3.6%**	**+9.1%**

(4) Comparison with existing smoothing methods.

Structure-preserving image smoothing has been studied in the image de-noising literature [118]. In this work, we have developed the SPLIC method for structure-preserving image completion. In the following experiment, we compare our SPLIC method with two existing image smoothing methods based on bilateral image filtering [119] and edge-guided image de-noising [118]. After processed by these methods, we retrain the network and evaluate the defense performance on the CIFAR-10 under white-box attacks. Figure 4.4 shows their classification accuracy comparison. We can see that, under FGSM, PGD and CW attacks, our method outperforms existing image smoothing methods. On the clean images, it can preserve the original image semantic structures much better than other methods. This is because our progressive smoothed rank function with total variation constraint can successfully capture and preserving the multi-scale semantic structures in the image, which are very important for network learning and image recognition.

4.4.3 Ablation Studies

In the following, we provide ablation studies to further understand our SPLIC method.

4.4.4 (1) Impact of the weighting parameter λ.

The SPLIC optimization problem in (4.5) has two objective functions weighted by the control parameter λ. Figure 4.5 shows the classification accuracy obtained by our SPLIC method with different λ on the CIFAR-10 dataset under white-box BPDA attack. We can see that the best performance is achieved for λ within the range of

$[0.01, 0.05]$. In our experiment, we set λ to be 0.02.

(2) Visualization of image samples in the learned feature space.

The proposed SPLIC method is able to remove local image texture details, encourage the network to focus on more discriminative semantic features, and improve the robustness of the network to adversarial noise. We expect that, in the learned feature space, images from different classes will have much better separation or larger margins since a small perturbation will not push the image sample across the decision boundary. To demonstrate this, we use the t-SNE method [120] to visualize the learned features on the CIFAR-10 dataset. Figure 4.6 (left) shows the visualization of image features without the SPLIC method. The right figure shows the result for the SPLIC method. We can see that, with the SPLIC pre-processing and training, images from different classes are much better separated, indicating significantly improved robustness against adversarial attacks.

(3) Performance on the clean images.

In this part, we compare the performance of our SPLIC algorithm on clean images with three low-rank approximation methods that have been implemented in [29] with source code publicly available. These methods are Universal Singular Value Thresholding (USVT), Soft Impute (Soft-Imp) and Nuclear Norm (NUC-Norm). The the classification accuracy are shown in Figure 4.7. We test these reconstruction methods with different percentages of pixels used for anchor points. We can see that, on the clean images without attack noise, our algorithm can maintain very good accuracy at different percentages of anchor points. This suggests that our method is able to

preserve the important semantic structures in the original images.

(4) Subjective examples of SPLIC results.

Figure 4.8 shows seven examples of SPLIC processing results. The first row shows the original images from CIFAR-10. The second row shows the attacked images by the white-box PGD method. The attack noise is clearly visible. The third row shows the restored images by our SPLIC method. We can see that the attack noise has been largely removed, the detailed image textures which are not important for recognition has been smoothed out, while important semantic structures are well preserved.

4.5 Conclusion

In this work, we observed that the adversarial attack operates at local image textures as a special noise while the human visual system focuses on semantic structures and global visual cues. Motivated by this, we developed a structure-preserving progressive low-rank image completion (SPLIC) method to remove unneeded texture details from the input images and let the deep neural network focuses more on global object structures and semantic cues. We formulate the problem into a low-rank matrix completion problem with progressively smoothed rank functions to avoid local minimums and total variation constraint to enforce local smoothness during the optimization process. Our experimental results demonstrate that the proposed method is able to successfully remove the insignificant local image details and let the network learning focus on global object structures. On black-box, gray-box, and white-box attacks, our method outperforms existing defense methods and significantly improves the ad-

versarial robustness of the network.

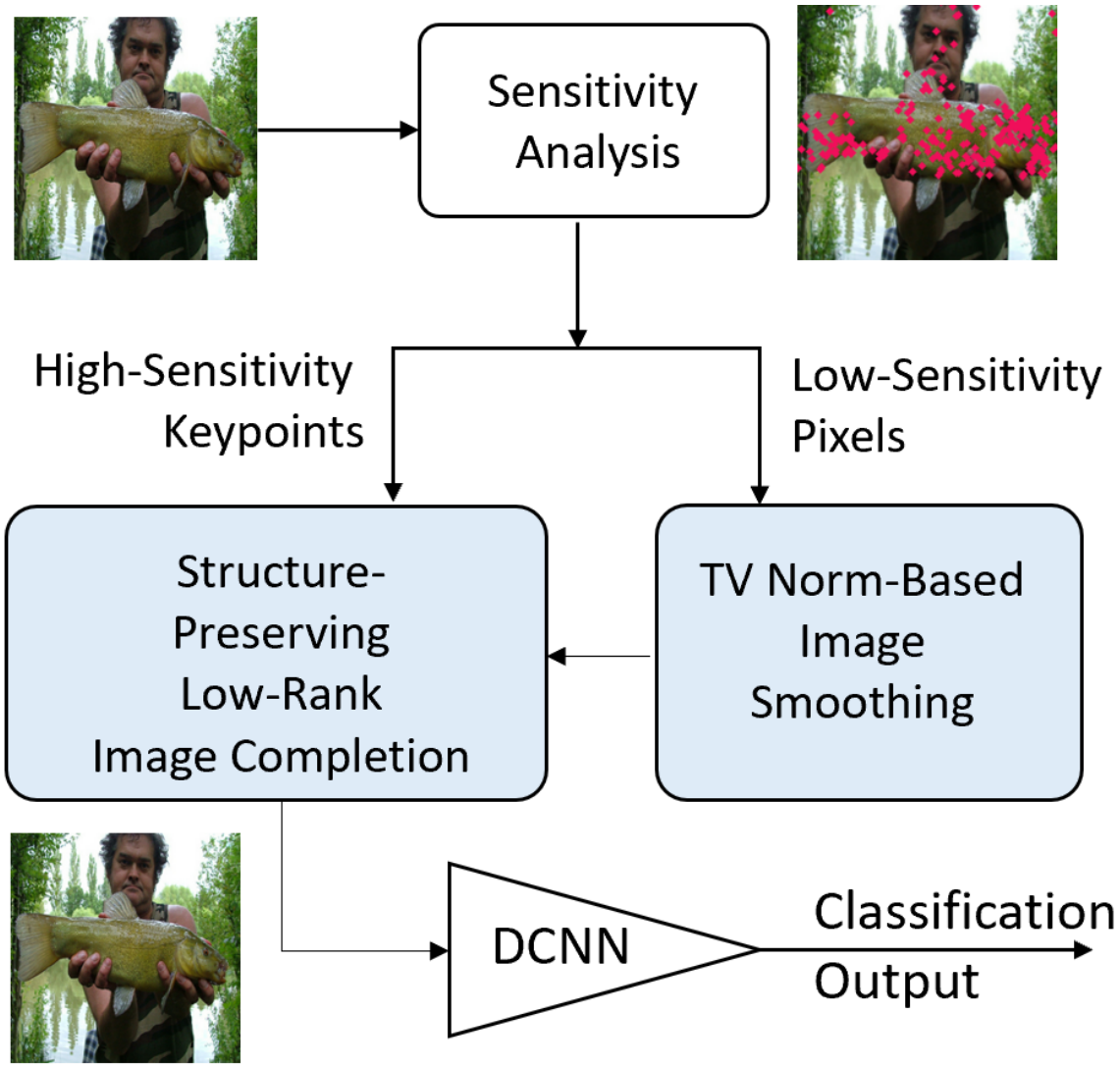


Figure 4.2: Illustration of the proposed structure-preserving low-rank image completion method for defending against adversarial attacks.

Figure 4.3: Example results by SPLIC: (a) the noise images and (b) the final SPLIC results.

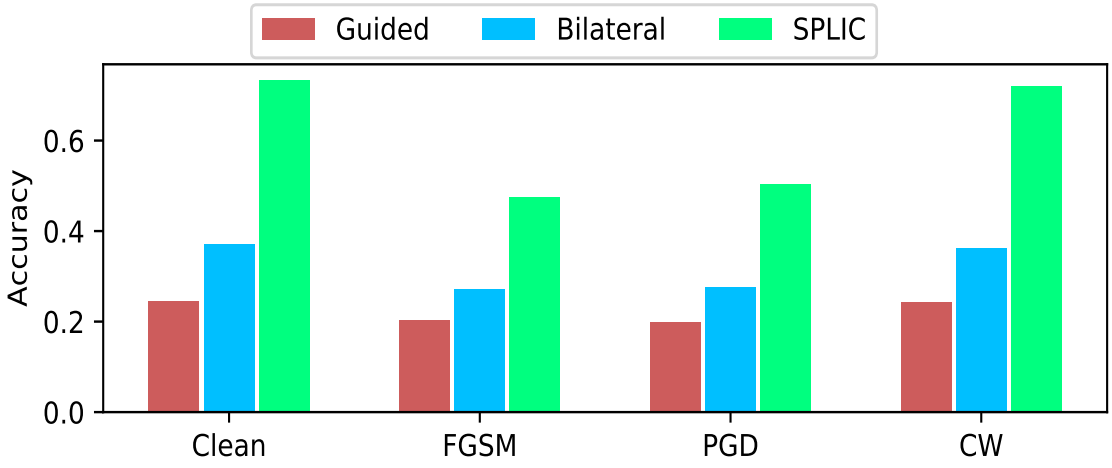


Figure 4.4: Comparison with existing image smoothing method.

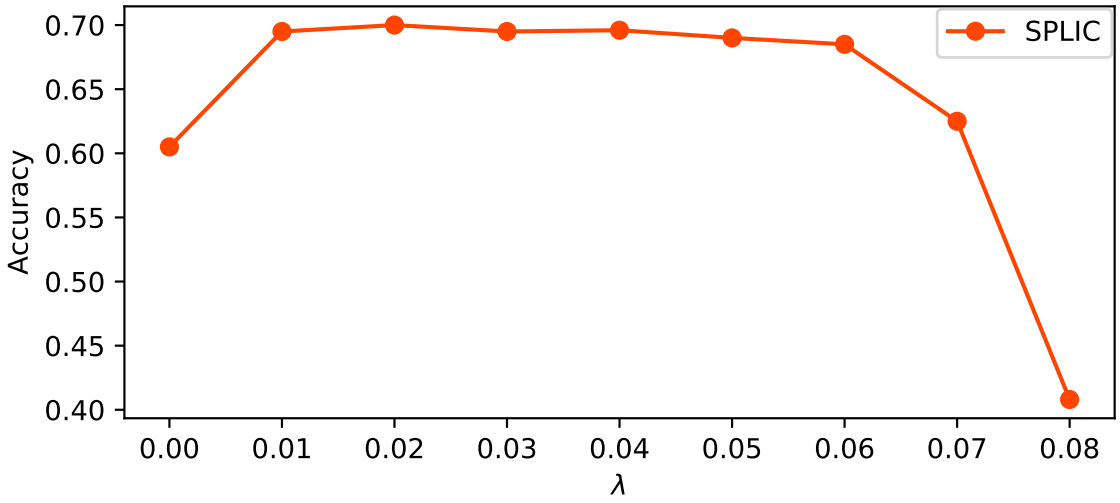


Figure 4.5: SPLIC accuracy with different λ.

Figure 4.6: Visualization of the samples in the learned feature space with the SPLIC method (right) and without defense (left).

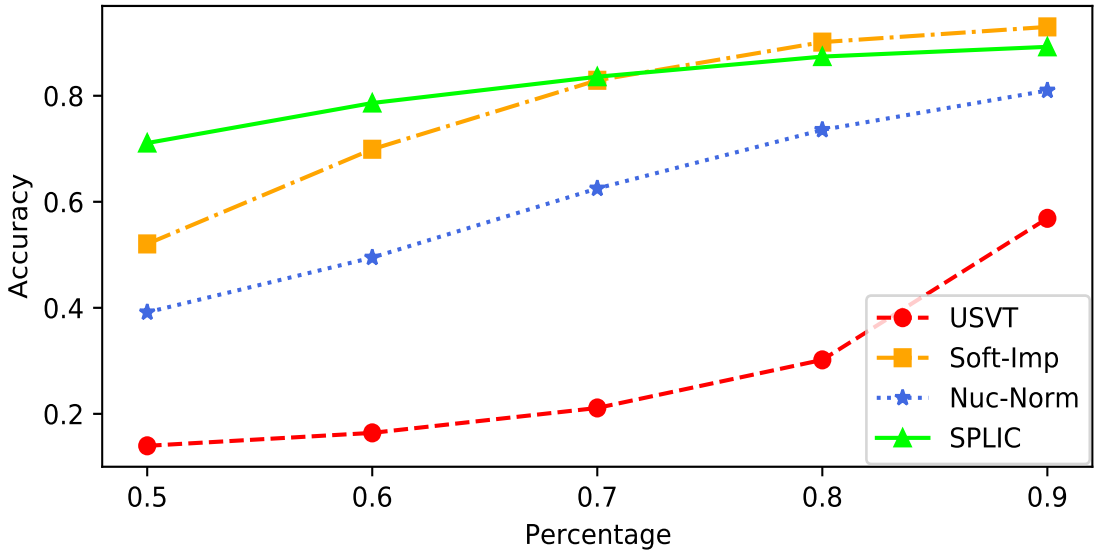


Figure 4.7: Reconstructions comparison on CIFAR-10 clean images with different probabilities of remained pixels.

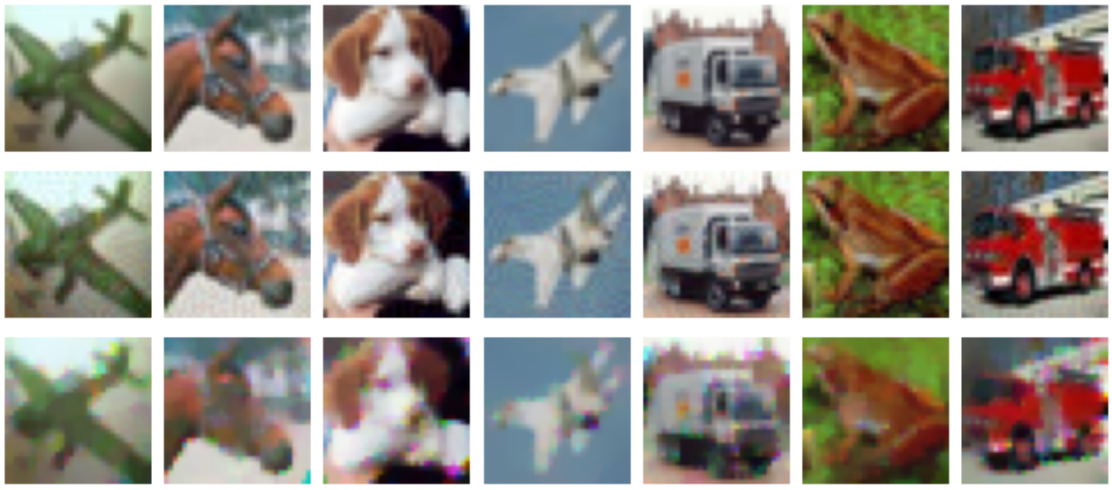


Figure 4.8: Image Samples: the first row are original images, the second row are adversarial images, and the last row are SPLIC images.

Bibliography

[1] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2018.

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *stat*, 1050:20, 2015.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[6] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[7] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[9] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[10] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[11] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pages 2074–2082, 2016.

[12] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):1943–1955, 2016.

[13] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[14] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018.

[15] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference. BMVA Press*, 2014.

[16] Wei Wen, Cong Xu, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Coordinating filters for faster deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 658–666, 2017.

[17] Alain Baccini, Ph Besse, and A Falguerolles. A l1-norm pca and a heuristic approach. *Ordinal and symbolic data analysis*, 1(1):359–368, 1996.

[18] Qifa Ke and Takeo Kanade. *Robust subspace computation using L1 norm*. School of Computer Science, Carnegie Mellon University Pittsburgh, PA, 2003.

[19] Qifa Ke and Takeo Kanade. Robust l_1 norm factorization in the presence of outliers and missing data by alternative convex programming. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 739–746. IEEE, 2005.

[20] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[21] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[22] Adith Boloor, Xin He, Christopher Gill, Yevgeniy Vorobeychik, and Xuan Zhang. Simple physical adversarial examples against end-to-end autonomous driving models. *arXiv preprint arXiv:1903.05157*, 2019.

[23] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519. ACM, 2017.

[24] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. In *International Conference on Learning Representations*, 2018.

[25] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

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

[27] Amir Beck and Marc Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE transactions on image processing*, 18(11):2419–2434, 2009.

[28] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[29] Yuzhe Yang, Guo Zhang, Dina Katabi, and Zhi Xu. Me-net: Towards effective adversarial robustness with matrix estimation. *arXiv preprint arXiv:1905.11971*, 2019.

[30] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[31] Qian Wang, Xianyi Zhang, Yunquan Zhang, and Qing Yi. Augem: automatically generate high performance dense linear algebra kernels on x86 cpus. In *SC'13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–12. IEEE, 2013.

[32] Vadim Lebedev and Victor Lempitsky. Fast convnets using group-wise brain damage. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2554–2564, 2016.

[33] Chenglong Zhao, Bingbing Ni, Jian Zhang, Qiwei Zhao, Wenjun Zhang, and Qi Tian. Variational convolutional neural network pruning. In *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2780–2789, 2019.

[34] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2019.

[35] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

[36] Hengyou Wang, Yigang Cen, Zhihai He, Ruizhen Zhao, Yi Cen, and Fengzhen Zhang. Robust generalized low-rank decomposition of multimatrices for image recovery. *IEEE Transactions on Multimedia*, 19(5):969–983, 2016.

[37] Hengyou Wang, Yigang Cen, Zhiquan He, Zhihai He, Ruizhen Zhao, and Fengzhen Zhang. Reweighted low-rank matrix analysis with structural smoothness for image denoising. *IEEE Transactions on Image Processing*, 27(4):1777–1792, 2017.

[38] Ian Jolliffe. Principal component analysis. In *International encyclopedia of statistical science*, pages 1094–1096. Springer, 2011.

[39] Matthew Partridge and Marwan Jabri. Robust principal component analysis. In *Neural Networks for Signal Processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop*, volume 1, pages 289–298. IEEE, 2000.

[40] John Wright, Arvind Ganesh, Shankar Rao, Yigang Peng, and Yi Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices

via convex optimization. In *Advances in neural information processing systems*, pages 2080–2088, 2009.

[41] Rui Ping Wen and Xi Hong Yan. A new gradient projection method for matrix completion. *Applied Mathematics and Computation*, 258:537–544, 2015.

[42] Xiaojie Guo and Zhouchen Lin. Low-rank matrix recovery via robust outlier estimation. *IEEE Transactions on Image Processing*, PP(99):1–1, 2018.

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[44] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014.

[45] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

[46] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[47] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[48] Mikhail Figurnov, Aizhan Ibraimova, Dmitry P Vetrov, and Pushmeet Kohli. Perforatedcnns: Acceleration through elimination of redundant convolutions. In *Advances in Neural Information Processing Systems*, pages 947–955, 2016.

[49] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018.

[50] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[51] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[52] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.

[53] John C Gower, Garmt B Dijksterhuis, et al. *Procrustes problems*, volume 30. Oxford University Press on Demand, 2004.

[54] Yoshio Takane and Heungsun Hwang. Regularized linear and kernel redundancy analysis. *Computational Statistics & Data Analysis*, 52(1):394–405, 2007.

[55] Yoshio Takane and Michael A Hunter. Constrained principal component analysis: a comprehensive theory. *Applicable Algebra in Engineering, Communication and Computing*, 12(5):391–419, 2001.

[56] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[57] J. Shi, W. Yang, and X. Zheng. Robust generalized low rank approximations of matrices. *Plos One*, 10(9):e0138028, 2015.

[58] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[59] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[60] Yin Zhang. Recent advances in alternating direction methods: Practice and theory. In *IPAM workshop on continuous optimization*, 2010.

[61] Jonathan Eckstein and Dimitri P Bertsekas. On the douglas—rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3):293–318, 1992.

[62] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[63] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[64] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[65] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[66] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proceedings of the IEEE conference on Symposium on Security and Privacy*, 2017.

[67] 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.

[68] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018.

[69] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[70] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147. ACM, 2017.

[71] Xiaojun Jia, Xingxing Wei, Xiaochun Cao, and Hassan Foroosh. Comdefend: An efficient image compression model to defend adversarial examples. *CoRR*, abs/1811.12673, 2019.

[72] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. *arXiv preprint arXiv:1812.03411*, 2018.

[73] Yulong Wang, Hang Su, Bo Zhang, and Xiaolin Hu. Interpret neural networks by identifying critical data routing paths. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8906–8914, 2018.

[74] Yuxian Qiu, Jingwen Leng, Cong Guo, Quan Chen, Chao Li, Minyi Guo, and Yuhao Zhu. Adversarial defense through network profiling based path extraction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4777–4786, 2019.

[75] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *Proceedings of the International Conference on Learning Representations*, 2018.

[76] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *Proceedings of the International Conference on Learning Representations*, 2018.

[77] Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proceedings of the International Conference on Machine Learning*, 2018.

[78] Eric Wong, Frank R. Schmidt, Jan Hendrik Metzen, and J. Zico Kolter. Scaling provable adversarial defenses. In *Proceedings of the Conference on Neural Information Processing Systems*, 2018.

[79] Xiao Zhang and David Evans. Cost-sensitive robustness against adversarial examples. In *Proceedings of the International Conference on Learning Representations*, 2019.

[80] E. J. Candès and Y. Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.

[81] Shiqian Ma, Donald Goldfarb, and Lifeng Chen. Fixed point and bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–353, 2009.

[82] Ran He, Tieniu Tan, and Liang Wang. Robust recovery of corrupted low-rank matrix by implicit regularizers. *IEEE Trans Pattern Anal Mach Intell*, 36(4):770–783, 2014.

[83] Hengyou Wang, Yigang Cen, Ruizhen Zhao, Viacheslav Voronin, Fengzhen Zhang, and Yanhong Wang. Fast smooth rank function approximation based on matrix tri-factorization. *Neurocomputing*, page S0925231217301558, 2017.

[84] Hengyou Wang, Ruizhen Zhao, and Yigang Cen. Rank adaptive atomic decomposition for low-rank matrix completion and its application on image recovery. *Neurocomputing*, 145:374–380, 2014.

[85] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[86] Hengyou Wang, Yigang Cen, Zhiquan He, Zhihai He, Ruizhen Zhao, and Fengzhen Zhang. Reweighted low-rank matrix analysis with structural smoothness for image denoising. *IEEE Transactions on Image Processing*, 27(4):1777, 2018.

[87] Yigang Peng, Jinli Suo, Qionghai Dai, and Wenli Xu. Reweighted low-rank matrix recovery and its application in image restoration. *IEEE transactions on cybernetics*, 44(12):2418–2430, 2014.

[88] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted l_1 minimization. *Journal of Fourier analysis and applications*, 14(5-6):877–905, 2008.

[89] Xiaoming Yuan and Junfeng Yang. Sparse and low-rank matrix decomposition via alternating direction methods. [Online], 2009.

[90] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[91] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[92] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018.

[93] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

[94] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[95] Yusuke Tsuzuku and Issei Sato. On the structural sensitivity of deep convolutional networks to the directions of fourier basis functions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 51–60, 2019.

[96] Yinpeng Dong, Hang Su, Baoyuan Wu, Zhifeng Li, Wei Liu, Tong Zhang, and Jun Zhu. Efficient decision-based black-box adversarial attacks on face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7714–7722, 2019.

[97] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*, 2018.

[98] Jonathan Uesato, Brendan O'Donoghue, Pushmeet Kohli, and Aäron van den Oord. Adversarial risk and the dangers of evaluating against weak attacks. In *ICML*, 2018.

[99] Bo Sun, Nian-hsuan Tsai, Fangchen Liu, Ronald Yu, and Hao Su. Adversarial defense by stratified convolutional sparse coding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11447–11456, 2019.

[100] Olga Taran, Shideh Rezaeifar, Taras Holotyak, and Slava Voloshynovskiy. Defending against adversarial attacks by randomized diversification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11226–11233, 2019.

[101] Edward Raff, Jared Sylvester, Steven Forsyth, and Mark McLean. Barrage of random transforms for adversarially robust defense. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6528–6537, 2019.

[102] Rajkumar Theagarajan, Ming Chen, Bir Bhanu, and Jing Zhang. Shieldnets: Defending against adversarial attacks using probabilistic adversarial robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6988–6996, 2019.

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

[104] Haichao Zhang and Jianyu Wang. Defense against adversarial attacks using feature scattering-based adversarial training. In *Advances in Neural Information Processing Systems*, pages 1829–1839, 2019.

[105] Shengyuan Hu, Tao Yu, Chuan Guo, Wei-Lun Chao, and Kilian Q Weinberger. A new defense against adversarial images: Turning a weakness into a strength. In *Advances in Neural Information Processing Systems*, pages 1635–1646, 2019.

[106] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Advances in Neural Information Processing Systems*, pages 3358–3369, 2019.

[107] Cihang Xie, Mingxing Tan, Boqing Gong, Alan Yuille, and Quoc V Le. Smooth adversarial training. *arXiv preprint arXiv:2006.14536*, 2020.

[108] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717, 2009.

[109] Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.

[110] Hooshang Ghasemi, Mohmmadreza Malek-Mohammadi, Massoud Babaie-Zadeh, and Christian Jutten. Srf: Matrix completion based on smoothed rank function. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3672–3675. IEEE, 2011.

[111] Mohammadreza Malek-Mohammadi, Massoud Babaie-Zadeh, Arash Amini, and Christian Jutten. Recovery of low-rank matrices under affine constraints via a smoothed rank function. *IEEE Transactions on Signal Processing*, 62(4):981–992, 2013.

[112] Hengyou Wang, Yigang Cen, Ruizhen Zhao, Viacheslav Voronin, Fengzhen Zhang, and Yanhong Wang. Fast smooth rank function approximation based on matrix tri-factorization. *Neurocomputing*, 257:144–153, 2017.

[113] Canyi Lu, Zhouchen Lin, and Shuicheng Yan. Smoothed low rank and sparse matrix recovery by iteratively reweighted least squares minimization. *IEEE Transactions on Image Processing*, 24(2):646–654, 2014.

[114] Adrian S Lewis. The convex analysis of unitarily invariant matrix functions. *Journal of Convex Analysis*, 2(1):173–183, 1995.

[115] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

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

[117] Chengzhi Mao, Ziyuan Zhong, Junfeng Yang, Carl Vondrick, and Baishakhi Ray. Metric learning for adversarial robustness. In *Advances in Neural Information Processing Systems*, pages 478–489, 2019.

[118] Kaiming He and Jian Sun. Fast guided filter. *arXiv preprint arXiv:1505.00996*, 2015.

[119] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*, pages 839–846. IEEE, 1998.

[120] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

VITA

Zhiqun Zhao was born in Shandong province in 1991. He received the B.S. degree in communication engineering from University of Electronic Science and Technology of China, UESTC, Chengdu, China, in 2013. Then he began to purchase M.S major in electronic and electrical engineering at Unversity of Missouri. He transferred to Ph.D. program working with Zhiha He since 2016 in Department of Electrical Engineering and Computer Science, University of Missouri. He has worked on many research projects on computer vision and low-rank matrix study. His current research interests include deep convolutional neural networks, low-rank approximation, adversarial examples and zero-shot learning.