TENSORSHIELD: Tensor-based Defense Against Adversarial Attacks on Images

Negin Entezari
University of California, Riverside
nente001@ucr.edu

Evangelos E. Papalexakis
University of California, Riverside
epapalex@cs.ucr.edu

***Abstract*—Recent studies have demonstrated that machine learning approaches like deep neural networks (DNNs) are easily fooled by adversarial attacks. Subtle and imperceptible perturbations of the data are able to change the result of deep neural networks. Leveraging vulnerable machine learning methods raises many concerns, especially in domains where security is an important factor. Therefore, it is crucial to design defense mechanisms against adversarial attacks. For the task of image classification, unnoticeable perturbations mostly occur in the high-frequency spectrum of the image. In this paper, we utilize tensor decomposition techniques as a preprocessing step to find a low-rank approximation of images that can significantly discard high-frequency perturbations. Recently a defense framework called SHIELD [1] could "vaccinate" Convolutional Neural Networks (CNN) against adversarial examples by performing random-quality JPEG compressions on local patches of images on the ImageNet dataset. Our tensor-based defense mechanism outperforms the SLQ method from SHIELD by 14% against Fast Gradient Descent (FGSM) adversarial attacks, while maintaining comparable speed.**

***Index Terms*—adversarial machine learning, deep neural networks, image classification.**

I. INTRODUCTION

In the last few years, Deep Neural Networks (DNNs) have been tremendously popular in various domains, including image processing and computer vision [2], [3], [4]. However, recently, the robustness of DNNs has been questioned when facing adversarial inputs. The performance of DNNs can significantly drop even on slightly perturbed instances [5]. For the task of image classification, attackers put constraints on perturbations such that they remain unnoticeable to the human eye, but they are still able to greatly deteriorate the performance of the model [6], [7], [8].

Utilizing machine learning methods that are vulnerable to adversarial attacks in a system where safety and security are critical factors may cause serious problems. Therefore, it is crucial to have a robust model against adversaries, especially in security-sensitive domains like autonomous driving and medical imaging. To address this concern, recent studies have researched to analyze the vulnerability of deep learning methods to come up with defense techniques against adversarial attacks [1], [9], [10], [11], [12].

To measure the strength of a perturbation, usually an l_2 or l_∞ norm is used. Adversarial perturbations are mostly designed to have a small norm and are unnoticeable to human inspection [13]. Designing a defense mechanism is a difficult task. Typically, the defender has only access to the perturbed instances (and definitely not the original ones, where there would be hope to identify which parts have been tampered with) and should be able to defend against different types of perturbations. Moreover, a defense mechanism specialized against a particular kind of attack could be easily defeated by new attacks which are optimized against its strategy. Therefore, designing a defense technique that captures a universal pattern across various attacks is highly desirable since this will allow to defend against most of the adversarial attacks.

SHIELD, proposed by Das et al. [1], is a real-time defense framework that performs JPEG compression with random levels over local patches of images to eliminate imperceptible perturbations, which mostly appear in the high-frequency spectrum of images. In this paper, we propose a tensor decomposition approach to compute a low-rank approximation of images that significantly discards high-rank perturbations. However, SHIELD considers images in isolation and ignores the correlation of images when facing adversarial attacks.

Our contributions are as follows:

- **Defense through the lens of factorization**: We propose a novel defense against adversarial attacks on images that utilizes tensor decomposition to reconstruct a low-approximation of perturbed images before feeding them to the deep network for classification. Without any retraining of the model, our method can significantly mitigate adversarial attacks.
- **Efficient and effective method**: Representing images with tensor allows processing images in batches as a 4-mode tensor, which is able to capture the latent structure of perturbations from multiple images rather than a single image and leads to more performance improvements.

II. RELATED WORK

A. Adversarial Attacks

We focus on defending against adversarial attacks on deep learning methods for the task of image classification. Here, we briefly outline some of the most popular attacks on images.

Given a classifier C, the goal of an adversarial attack is to modify an instance x to a perturbed instance x' such that $C(x) \neq C(x')$, while keeping the distance $\|x - x'\|$ between perturbed and clean instance small [14]. By $\|.\|$ we denote some norm to express the strength of the perturbations. The popular choices are Euclidean distance (l_2 norm) and

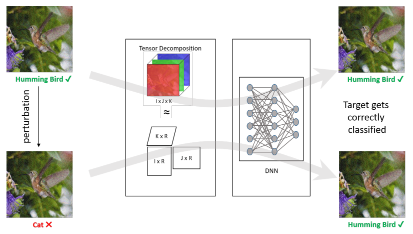

Fig. 1: System Overview: low-rank tensor approximation of images to "vaccinate" the network against perturbations. (the term "vaccinate" was first used by Das et al. [1] to refer to models equipped with a defense mechanism.)

Chebyshev distance (l_∞ norm). Here, we discuss some popular attacks against which we evaluate our proposed method.

Fast Gradient Sign Method (FGSM)[7]: FGSM is a fast method to compute perturbations that is based on computing first-order gradients. FGSM generates adversarial images by introducing a perturbation as follows:

$$x' = x + \epsilon \ \text{sgn}(\nabla J_x(\theta, x, y)) \tag{1}$$

where ϵ is a user-defined threshold that determines the strength of the perturbations and controls its magnitude per pixel. θ is the parameter of the model, y is the true label of the instance x, and J is the cost of training the neural network.

Iterative Fast Gradient Sign Method (I-FGSM)[15]: I-FGSM is the iterative version of the FGSM. In each iteration i, I-FGSM clips the pixel values to remain within the l_∞ neighborhood of the corresponding values from a "clean" instance x:

$$x'_i = x'_{i-1} + \alpha \ \text{sgn}(\nabla J_{x'_{i-1}}(\theta, x'_{i-1}, y)) \tag{2}$$

Projected Gradient Descent (PGD)[16]: PGD is one of the strongest gradient-based attacks [16] Given a clean image x, PGD aims to find a small perturbation $\delta \in \mathcal{S}$ to generate the perturbed instance $x' = x + \delta$. PGD starts from a random perturbation and iteratively updates the perturbation:

$$\delta_i = \Pi_{\mathcal{S}}[\delta_{i-1} + \tau \ \text{sgn}(\nabla_x L(x + \delta_{i-1}, y))] \tag{3}$$

where τ is a fixed step size. $\Pi_{\mathcal{S}}$ projects the perturbation onto set $\mathcal{S}$, set of allowed perturbations in the ϵ neighborhood the "clean" instance x.

B. *Defense Against Adversarial Attacks*

Defense mechanisms against adversaries fall into two main categories [14]. The first group of methods aim to train the classifier on adversarial examples to make it robust against them [17], [7], [18]. This approach requires extensive training on various adversarial examples. The second group of defenses try to remove perturbations from the adversarial example by performing some transformation [1], [19], [9], [19], [20]. These transformed examples are then fed to the model, and the objective is to achieve the same classification as the clean instance. In this paper, our focus is on the second approach.

SHIELD proposed by Das et al. [1], uses image preprocessing as a defense mechanism to reduce the effect of perturbations. SHIELD is based on the observation that the attacks are high-frequency. Thus, eliminating those high-frequencies (which are not generally visible by the human eye) will sanitize the image. SHIELD performs Stochastic Local Quantization (SLQ) as a preprocessing step and subsequently employs JPEG compression with qualities 20, 40, 60, and 80 on the image, then for each 8×8 block of the image, randomly selects from one of the compressed images.

Variational Autoencoders (VAEs) [21], [22] have been used to defend against adversaries on images. In a study by Luo et. al. [23], they utilize VAE to map high-dimensional images to a lower-dimensional latent space in which most of the perturbation is discarded. Autoencoders [24] are a type of neural networks that learn a low-dimensional representation of the data in an unsupervised manner and variational autoencoders incorporate random sampling in the encoding and decoding process. The low-rank representation of images learned by the autoencoder can improve the performance of the deep model when facing adversarial examples. However, the drawback of autoencoder defenses is that they need to be trained on the data, which makes them not suitable for online applications.

In this paper, we preprocess images using tensor decomposition techniques to achieve a low-rank approximation of the image. We can significantly alleviate the effect of perturbations without performing any training on the dataset. In a parallel approach, [25] employs singular value decomposition to compute a low-rank approximation of graph to defend against adversarial attacks on graphs. However, this paper is the first to identify and leverage the observation that gradient-based attacks on deep learning image classifiers are manifested in high-rank components of a decomposition of the image.

III. Proposed Method

In this section, we first investigate the characteristics of adversarial attacks on networks designed for the task of image classification. Then we propose a tensor-based defense mechanism against these attacks, which improves the performance of the network.

A. *Characteristics of Image Perturbations*

Assume a trained model C with high accuracy on clean images is given. Adversarial attacks perform perturbations on the clean images in a way that they are imperceptible to humans yet are successful in deceiving the model to misclassify the perturbed instances. In other words, for a clean image x and its corresponding perturbed image x', the goal is to have: $C(x) \neq C(x')$. The adversarial attacks do not preserve the spectral characteristics of images and add high-frequency components to images to remain unnoticeable to the human eyes [1]. Perturbations in the image domain are crafted in a way that mostly affect the high-frequency spectrum of images. Therefore, discarding the high-frequency factors of the image using approaches like compression or low-rank approximation of images could be successful defenses against these types of perturbations. Therefore, a mechanism that only keeps the low-rank components of the image and discards the high-rank ones can be successful in discarding the perturbations. In [1], the authors leverage JPEG compression to remove high-frequency components of the image and alleviate the effect of perturbations. In this paper, we study the problem from a "matrix spectrum" point of view (i.e., the singular value profile and the intrinsic low-rank dimensionality of the data) and use tensor decomposition techniques to achieve a low-rank approximation of perturbed images.

B. ***TensorShield**: Tensor-based Defense Mechanism*

We briefly explain the concepts and notations used.

A tensor, denoted by $\underline{\mathbf{X}}$, is a multidimensional matrix. The order of a tensor is the number of modes/ways and is the number of indices required to index the tensor [26], [27]. An RGB image is a three-mode tensor where the first and second modes correspond to the pixels and the third mode corresponds to the red, green, and blue channels, i.e. the frontal slices are red, green, and blue channels of the image. An RGB image of size $W \times H$ is a 3-mode tensor of size $W \times H \times 3$, where W and H are the width and height of the image, respectively.

To achieve a low-rank approximation of the perturbed images, we perform a tensor decomposition technique on the image and by choosing small values for the rank of the tensor, we reconstruct a low-rank approximation of the image which is fed to the deep network. The low-rank approximation of image discards high-frequency perturbations which can improve the performance of the network on the perturbed images. However, traditional tensor decomposition techniques like CP/Parafac [28] and Tucker[29] are time-consuming and may slow down the neural network performance which makes our proposed method impractical for real-time defense. To overcome this issue, we leverage Tensor-Train decomposition [30] which scales linearly with respect to the dimension of the tensor and was especially introduced to address the problem of curse of dimensionality [30]. This highly-desirable property of the Tensor-Train allows us to process images in batches that form a 4-mode tensor and perform the Tensor-Train decomposition on 4-mode tensors quite fast. For a batch of N images, the size of the 4-mode tensor will be $N \times W \times H \times 3$. Generally, decomposing a 4-mode tensor is slower compared to a 3-mode one. However, by considering images in batches, some of the I/O overhead is reduced, that results in almost the same processing time on the entire dataset. Furthermore, processing images in batches improves the performance of the model, because decomposing images in batches extracts latent structure corresponding to perturbations from multiple images and captures general characteristics of perturbations.

For a 4-mode tensor, the Tensor-Train decomposition can be written as follows:

$$\underline{\mathbf{X}}(i,j,k,l) \approx \sum_{r_1,r_2,r_3} \mathbf{G}_1(i,r_1)\underline{\mathbf{G}}_2(r_1,j,r_2)\underline{\mathbf{G}}_3(r_2,k,r_3)\mathbf{G}_4(r_3,l) \quad (4)$$

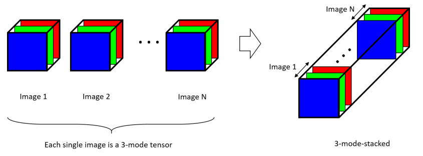

Fig. 2: Stacking 3-mode images along the third mode.

Another possible representation for the batch of images is to convert the 4-mode tensor to a 3-mode tensor by stacking the images along the third mode, i.e., stacking RGB channels and the result tensor will be of dimension $W \times H \times 3 * N$. Fig. 2 illustrates a 3-mode stacked tensor of N images. There are other ways to convert a 4-mode tensor into a 3-mode one. For instance, another way is to flatten the RGB image into a matrix with three columns corresponding to the channels of the image. With this representation, the final tensor will be of size $W * H \times 3 \times N$. One disadvantage of this representation is that flattening the image ignores the spatial relationship of the pixels. Moreover, with this vectorized representation, the first dimension is much bigger than the other two dimensions and requires a larger value of rank to get a reasonable approximation of the image, and larger ranks make the decomposition slower. For these reasons, we do not consider the vectorized representation in our study. In the experimental evaluations that follow, we will examine different representation including a single image versus batches of images and 3-mode tensors versus 4-mode tensors.

IV. Experimental Evaluation

In this section, we show how the proposed method can successfully remove adversarial perturbations and we compare our results to SHIELD (SLQ). According to [31], original SHIELD evaluations have gained benefit from central cropping of images in their evaluations, whereas the perturbations were generated with cropping being off. In all our evaluations, we disable the central cropping.

A. *Experiment Setup*

We performed experiments on the validation set of the *ImageNet* dataset, which includes 50,000 images from 1,000 classes [32]. All experiments are performed on the *ResNet-v2 50* model [2] from the *TF-Slim* module of *TensorFlow* [33]. The adversarial attacks are from the CleverHans package [1] [34]. We performed the experiments on a machine with one NVIDIA Titan Xp (12 GB) GPU. We used TensorLy [2] library in Python to perform tensor decomposition techniques [35].

B. *Parameter Tuning*

In our evaluations, we express different configurations in the form of a list as [tensor decomposition, tensor representation, batch size, rank] and we investigate the accuracy and runtime of the ResNet-v2 50 on 1000 images from the ImageNet

[1]https://github.com/tensorflow/cleverhans
[2]https://github.com/tensorly/tensorly

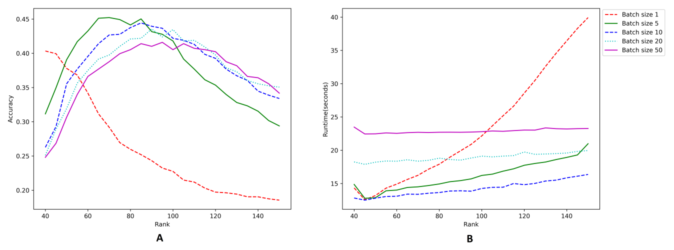

Fig. 3: Accuracy (A) and runtime (B) of ResNet-v2 50 over 1000 images attacked by FGSM ($\epsilon = 4$). Tensor-Train decomposition is applied on a single image (batch size 1) or 4-mode tensor of batches of size 5, 10, 20, and 50 to defend against FGSM perturbations.

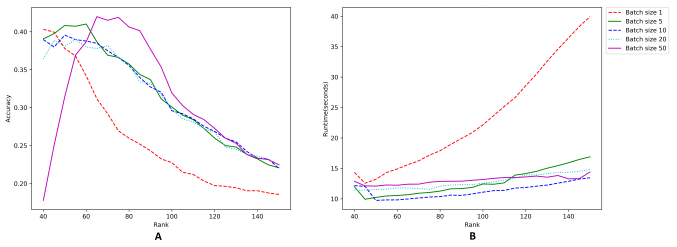

Fig. 4: Accuracy (A) and runtime (B) of ResNet-v2 50 over 1000 images attacked by FGSM ($\epsilon = 4$). Tensor-Train decomposition is applied on a single image (batch size 1) or 3-mode-stacked tensor of batches of size 5, 10, 20, and 50 to defend against FGSM perturbations.

dataset for different configurations. The possible values for each part of the configuration list is as follows:

- Tensor decomposition: {Parafac, Tucker, Tensor-Train}
- Tensor repres.: {3-mode, 3-mode-stacked, 4-mode}
- Batch size: {1, 5, 10, 20, 50}
- Rank: varies by choice of repres. and decomposition.

Performing tensor decomposition for a batch of images can reduce the decomposition overhead compared to decomposing a single image and accelerates the entire evaluation process. Moreover, considering images in batches helps to better capture the pattern of perturbations from multiple images. However, the choice of the right batch size is important. A large batch of images needs larger ranks for decomposition and could get very slow. Also, in a large batch of images, the variety of images that are from different classes increases which deteriorates the performance of the decomposition. To find the best batch size, we perform a grid search on values 5, 10, 20, and 50. Tensor Train decomposition of a 4-mode tensor requires setting 3 values for the ranks. The first value corresponds to compressing the batches, the second value corresponds to compressing the image pixels, and the third value corresponds to compressing the RGB channels. We fix the first rank to the number of batches and the third rank to the number of channels i.e., 3. For the second rank, we search within the range 40 to 150. Fig. 3 shows the accuracy and runtime of the model for different batch sizes for Tensor-Train decomposition with ranks ranging from 50 to 120 with steps of 5. The figure also shows how processing single images (batch size 1) differs from batch sizes greater than 5. In the case that we are processing single images, the runtime increases as the rank gets larger, however, as the batch size increase, the runtime becomes less sensitive to the ranks and for the batch size 50 it will become almost constant for all the ranks. Batch size 5 produces the highest accuracy, while batch size 10 has the lowest runtime. There is a trade-off between runtime and accuracy. Based on the priorities of the system, one might sacrifice accuracy for speed. Fig. 4 shows the effect of different batch sizes on the 3-mode-stacked representation. Plots for batch sizes 5, 10, and 20 are almost identical in both accuracy and runtime. Batch size 50 produces the curacy with the 3-mode-stacked representation. However, the highest accuracy with the 3-mode-stacked representation is lower than the highest accuracy achieved using the 4-mode representation.

C. Results

As mentioned in Section III, Tensor-Train performs much faster than Parafac and Tucker. Therefore, for the Parafac and Tucker, we only report the result for the configuration which corresponds to the maximum accuracy as a reference for comparison against Tensor-Train.

*1) **Comparing Against SLQ on ImageNet**:* We compare the performance of *ResNet-v2 50* model when vaccinated using SLQ and *TensorShield* methods. PGD, FGSM, and I-FGSM attacks are used to generate adversarial examples. Table I summarizes the result. Different configurations of *TensorShield* are reported in the table.

As illustrated in Table I, Tensor-Train outperforms Tucker and Parafac with respect to both accuracy and runtime. Tensor-Train performed on 4-mode tensor has produced the highest accuracy. As explained earlier, processing images in batches better captures latent components corresponding to perturbation by leveraging higher-order correlations. Tensor-Train can be utilized with different tensor representations (3-mode, 3-mode-stacked, or 4-mode) to adjust to needs for higher accuracy or higher speed. While the 4-mode representation produces the highest accuracy, the 3-mode single image representation can be used to speed up the process, with a small drop in the accuracy. SLQ is the fastest among all defenses, but it has the lowest accuracy.

*2) **Comparing Against VAE on MNIST**:* Here, we compare our method against the variational autoencoder defense proposed in [23] on the MNIST dataset. Defense mechanisms based on variational autoencoders require extensive training and hence they are not easily scalable to larger datasets. Therefore, we compared our method against VAE defense on the small MNIST dataset. CNN model used for the image classification has the same architecture as explained in [23]. The CNN model contains 2 convolutional layers followed by a fully-connected layer. Both of the convolutional layers use 5×5 filter size with 32 and 64 channels in the first and second convolutional layers, respectively. After each convolutional layer, 2×2 max-pooling with stride 2 is used.

TABLE I: Summary of accuracies and runtime of ResNet-v2 50 on ImageNet validation set against FGSM, I-FGSM, and PGD adversarial attacks for defenses with different configurations.

Configurations	**PGD** ($\epsilon = 4$)	**FGSM** ($\epsilon = 4$)	**I-FGSM** ($\epsilon = 4$)	**Runtime** (seconds)
No defense	11.10	18.40	7.49	
[Tensor-Train, 4-mode, 5, [5,90,3]]	**51.53**	**43.59**	**50.46**	675
[Tensor-Train, 4-mode, 10, [10,100,3]]	51.01	43.10	49.95	605
[Tensor-Train, 3-mode, 1, 40]	49.75	42.32	48.52	**530**
[Tucker, 3-mode-stacked, 30, [105,105,90]]	49.37	40.07	48.79	1050
[Parafac, 3-mode, 1, 60]	48.11	41.38	49.75	5500
SLQ	44.60	29.40	38.60	**410**

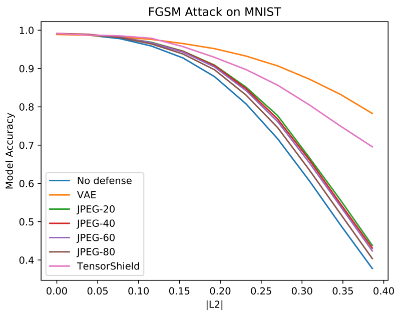

Fig. 5: FGSM attack on CNN model for MNIST dataset. For small and less noticeable perturbations, *TensorShield* outperforms other defenses, however as perturbations get stronger, VAE outperforms our method. However, VAE requires extensive training which makes it less applicable to large datasets, whereas *TensorShield* does not require any training on the dataset and can be applied on large datasets.

The fully-connected layer has 1024 hidden nodes that uses Relu activation with a dropout rate of 0.4.

Table II summarizes the performance of different defenses including *TensorShield*, VAE, and JPEG compression with qualities 20,40,60, and 80. For *TensorShield* we only report the configuration with the best accuracy which is a 3-mode tensor containing batches of 5 images ($28 \times 28 \times 5$). We used Tensor-Train decomposition with rank 30. Size of MNIST images is 28×28 and a tensor including a single image will be a 2-dimensional array and a tensor composing a batch of N images will be a 3-mode tensor of size $28 \times 28 \times N$. Fig. 5 shows the performance of *TensorShield* and VAE against the FGSM attack. Also different quality of JPEG compression was applied on images as a preprocessing step to resist against adversaries. For small, less noticeable perturbations, *TensorShield* is on par or better than VAE, but *TensorShield* has a higher drop in performance for stronger attacks. Even for strong attacks, the performance improvement achieved by *TensorShield* is significant and due to its scalability, *TensorShield* is a feasible option for vaccinating models on large datasets.

*3) **Introducing Randomness to the Defense Framework**:* Incorporating randomness in the defense framework makes the job of the attacker more difficult to deal with a random strategy rather than a fixed one. By selecting randomly from a set of ranks, we can add randomness to the tensor decomposition process. Another way is to split images into small patches, similar to local 8×8 patches from SHIELD, and perform decomposition of a random rank on each patch and stitch up the patches to reconstruct a randomized low-rank approximation of images. In a 4-mode tensor representation, splitting images into patches creates smaller 4-mode tensors, e.g. splitting a 4-mode tensor including 5 batches of images with size $300 \times 300 \times 3$ into patches of size 50×50 creates 6 tensors of size $5, 50, 50, 3$. Table III shows the results of incorporating

TABLE II: Summary of accuracies of CNN model on MNIST dataset against PGD, FGSM, and I-FGSM adversarial attacks. The performance of the model is reported when it is vaccinated using different defenses including *TensorShield*, VAE, and JPEG compression with qualities 20, 40, 60, and 80.

Defense	**No attack**	**PGD** ($\epsilon = 4$)	**FGSM** ($\epsilon = 4$)	**I-FGSM** ($\epsilon = 4$)
No defense	99.12	97.85	97.80	97.50
JPEG-20	99.12	98.28	98.13	98.20
JPEG-40	99.12	98.35	98.20	98.27
JPEG-60	99.12	98.29	98.17	98.28
JPEG-80	98.11	98.25	98.09	98.10
VAE	98.87	98.30	98.23	98.18
TensorShield	99.12	**98.44**	**98.32**	**98.31**

TABLE III: Accuracies and runtime of ResNet-v2 50 on ImageNet validation set against PGD, FGSM, and I-FGSM adversarial attacks with $\epsilon = 4$ vaccinated using Tensor-Train with 4-mode tensor of batch size 5. Decomposition rank is randomly selected from a set of possible ranks. No patching is equivalent to full size image.

Patch size	Ranks	PGD ($\epsilon = 4$)	FGSM ($\epsilon = 4$)	I-FGSM ($\epsilon = 4$)	Runtime (seconds)
[8,8]	[5,3,3] [5,5,3] [5,7,3]	47.89	41.04	48.26	2550
[50,50]	[5,10,3] [5,20,3] [5,30,3]	48.35	41.97	48.12	1100
[150,150]	[5,40,3] [5,50,3] [5,60,3] [5,70,3]	50.96	42.12	48.98	765
None	[5,70,3] [5,90,3] [5,110,3]	50.48	42.73	49.68	710

randomness with tensor decomposition. With smaller patch sizes, the decomposition will have more overhead, hence the runtime complexity increases.

V. Conclusions

In this paper, we explored the efficacy of low-rank tensor decomposition of perturbed images during the preprocessing step in helping to defend against adversarial attacks. The low-rank approximation of perturbed images was fed to the deep network for the task of classification which could significantly improve the performance of the model. We evaluated our method against popular adversarial attacks: FGSM, I-FGSM, and PGD. We also compared our method against two different defense mechanisms: SLQ and VAE where the former uses JPEG compression as defense and the latter learns a low-rank representation of images using variational autoencoders. We illustrated that considering images in small batches better captures the latent structure of perturbations and helps to improve the performance of the model. We also showed how different configurations of batch sizes and decomposition ranks, allow to trade-off between accuracy and runtime.

Acknowledgment

This research was sponsored by the U.S. Army Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Combat Capabilities Development Command Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, S. Li, L. Chen, M. E. Kounavis, and D. H. Chau, "Shield: Fast, practical defense and vaccination for deep learning using jpeg compression," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 196–204.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

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

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

[7] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[8] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, 2018.

[9] A. N. Bhagoji, D. Cullina, and P. Mittal, "Dimensionality reduction as a defense against evasion attacks on machine learning classifiers," *arXiv preprint*, 2017.

[10] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," *arXiv preprint arXiv:1702.04267*, 2017.

[11] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016.

[12] C. Xie, Y. Wu, L. v. d. Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 501–509.

[13] B. Luo, Y. Liu, L. Wei, and Q. Xu, "Towards imperceptible and robust adversarial example attacks against neural networks," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[14] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

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

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

[17] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári, "Learning with a strong adversary," *arXiv preprint arXiv:1511.03034*, 2015.

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

[19] C. Guo, M. Rana, M. Cisse, and L. Van Der Maaten, "Countering adversarial images using input transformations," *arXiv preprint arXiv:1711.00117*, 2017.

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

[21] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[22] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[23] Y. Luo and H. Pfister, "Adversarial defense of image classification using a variational auto-encoder," *arXiv preprint arXiv:1812.02891*, 2018.

[24] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML workshop on unsupervised and transfer learning*, 2012, pp. 37–49.

[25] N. Entezari, S. Al-Sayouri, A. Darvishzadeh, and E. Papalexakis, "All you need is low (rank): Defending against adversarial attacks on graphs," in *13th ACM International Conference on Web Search and Data Mining WSDM*, 2020.

[26] T. Kolda and B. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, 2009.

[27] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "Tensors for data mining and data fusion: Models, applications, and scalable algorithms," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 2, p. 16, 2017.

[28] R. Harshman, "Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis," 1970.

[29] L. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[30] I. V. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[31] C. Cornelius, "The efficacy of shield under different threat models," *arXiv preprint arXiv:1902.00541*, 2019.

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[33] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[34] N. Papernot, N. Carlini, I. Goodfellow, R. Feinman, F. Faghri, A. Matyasko, K. Hambardzumyan, Y.-L. Juang, A. Kurakin, and R. Sheatsley, "cleverhans v2. 0.0: an adversarial machine learning library," *arXiv preprint arXiv:1610.00768*, 2016.

[35] J. Kossaifi, Y. Panagakis, A. Anandkumar, and M. Pantic, "Tensorly: Tensor learning in python," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 925–930, 2019.