SPARSE CODING FRONTEND FOR ROBUST NEURAL NETWORKS

Can Bakiskan, Metehan Cekic, Ahmet Dundar Sezer, Upamanyu Madhow
Department of Electrical and Computer Engineering
University of California Santa Barbara
Santa Barbara, CA, 93106, USA
{canbakiskan,metehancekic,adsezer,madhow}@ece.ucsb.edu

ABSTRACT

Deep Neural Networks are known to be vulnerable to small, adversarially crafted, perturbations. The current most effective defense methods against these adversarial attacks are variants of adversarial training. In this paper, we introduce a radically different defense trained only on clean images: a sparse coding based frontend which significantly attenuates adversarial attacks before they reach the classifier. We evaluate our defense on CIFAR-10 dataset under a wide range of attack types (including ℓ^∞, ℓ^2, and ℓ^1 bounded attacks), demonstrating its promise as a general-purpose approach for defense.

1 INTRODUCTION

Deep neural networks (DNNs) are known to be vulnerable to small, adversarially designed perturbations (Biggio et al., 2013; Szegedy et al., 2014). Since the discovery of such adversarial attacks, researchers have tried various defense strategies such as adversarial training (Madry et al., 2018; Zhang et al., 2019), constraining Lipschitz constant (Cisse et al., 2017), randomized smoothing (Cohen et al., 2019), and preprocessing methods (Guo et al., 2017; Yang et al., 2019). Sadly, the majority of the proposed defenses have been subsequently defeated (Carlini & Wagner, 2017; Athalye et al., 2018). Current state-of-the-art defenses, as determined by several independent research efforts, are variants of adversarial training (Zhang et al., 2019; Carmon et al., 2019).

In this paper, we propose, and extensively evaluate on the CIFAR-10 image dataset, a novel preprocessing defense trained only on clean images. We employ sparse coding using an overcomplete patch-level dictionary, and mitigate adversarial perturbations by (1) zeroing out all but the top T coefficients of the basis projection, thus restricting the attack space, and (2) quantizing the surviving coefficients to ± 1, thus not allowing perturbations to ride on top of them. The defense parameters are chosen to combat ℓ^∞ bounded attacks, but we demonstrate its general-purpose nature by showing that it also provides robustness against ℓ^2 and ℓ^1 bounded attacks. The code for our defense can be found at: `https://github.com/canbakiskan/sparse_coding_frontend`

2 ADVERSARIAL ATTACKS

Whitebox attacks assume that the adversary has access to the model and its weights. For typical ℓ^p bounded attack models, projected gradient descent (PGD) is currently known to be the strongest. The ℓ^p budget ϵ is divided into smaller steps δ. In each step, gradient of the loss function $\mathcal{L}$ with respect to the input is normalized and scaled by δ, and added to the perturbation calculated in the previous step. After each step, the computed attack is clipped to conform to the ℓ^p bound. Mathematically, PGD can be written as

$$\mathbf{e}_{i+1} = \text{clip}_\epsilon^p\big[\mathbf{e}_i + \delta \cdot \frac{\nabla_\mathbf{e}\mathcal{L}(\mathbf{f}(\mathbf{x}+\mathbf{e}_i),\mathbf{y})}{||(\nabla_\mathbf{e}\mathcal{L}(\mathbf{f}(\mathbf{x}+\mathbf{e}_i),\mathbf{y})||_p}\big] \tag{1}$$

where $\mathbf{x}$ is the input, $\mathbf{y}$ is the target output, and $\mathbf{e}_i$ is the adversarial perturbation calculated in step i. The loss function $\mathcal{L}$ is usually taken as the cross-entropy loss between the network outputs $\mathbf{f}(\mathbf{x})$ and target output $\mathbf{y}$. Through equation 1, gradient ascent over the loss in the input space is achieved.

To optimize over a greater area in the input space, the attack can be initialized with random restarts within the ϵ ℓ^p ball of the clean input. After the computation of equation 1 for each restart, a restart that successfully fools our model is chosen.

A modified version of PGD is called Carlini-Wagner (C&W) attack. In this attack, the loss function $\mathcal{L}$ is taken as the difference between the logit that corresponds to the correct label and the maximum of logits among the incorrect labels.

Blackbox attacks do not assume full knowledge of the model and weights. Decision-based black box models assume access to the decision outputs of the model (Brendel et al., 2017), while purely blackbox models assume only access to the training dataset, typically using a surrogate model (transfer blackbox) trained on the same dataset in order to generate attacks (Tramèr et al., 2017).

3 FRONTEND

We consider RGB images of size $N \times N \times 3$, processed using $n \times n \times 3$ patches with stride S, so that we process $M = m \times m$ patches, where $m = \lfloor (N-n)/S \rfloor + 1$. We learn a patch-level overcomplete dictionary (number of basis tensors much larger than the patch dimension $3n^2$) in unsupervised fashion, and use it to extract sparse local features. These outputs go through a CNN "decoder" whose goal is to restore the output dimension to $N \times N \times 3$ rather than to reconstruct the image. This frontend is followed by a standard CNN classifier, trained jointly with the decoder using standard supervised learning.

3.1 PATCH-LEVEL OVERCOMPLETE DICTIONARY

We use a standard algorithm (Mairal et al., 2009) (implemented in Python library `scikit-learn`) which is a variant of K-SVD (Elad & Aharon, 2006). Given a set of clean training images $\mathcal{X} = \{\mathbf{X}^{(k)}\}_{k=1}^{K}$, an overcomplete dictionary $\mathbf{D}$ with L atoms is learnt by solving the following optimization problem (Mairal et al., 2009):

$$\min_{\mathbf{D}\in\mathcal{C},\{\boldsymbol{\alpha}^{(k)}\}_{k=1}^{K}} \sum_{k=1}^{K}\sum_{i,j}\Big(\frac{1}{2}\left\|\mathbf{R}_{ij}\mathbf{X}^{(k)} - \mathbf{D}\boldsymbol{\alpha}_{ij}^{(k)}\right\|_2^2 + \lambda\left\|\boldsymbol{\alpha}_{ij}^{(k)}\right\|_1\Big) \tag{2}$$

where $\mathcal{C} \triangleq \big\{\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_L] \in \mathbb{R}^{\bar{n}\times L} \mid \|\mathbf{d}_l\|_2 = 1, \forall l \in \{1,\ldots,L\}\big\}$, λ is a regularization parameter, $\boldsymbol{\alpha}^{(k)}$ is an $m \times m \times L$ tensor containing the coefficients of the sparse decomposition, and $\mathbf{R}_{ij} \in \mathbb{R}^{\bar{n}\times\bar{N}}$ with $\bar{n} \triangleq 3n^2$ and $\bar{N} \triangleq 3N^2$ extracts the (ij)-th patch from image $\mathbf{X}^{(k)}$. The optimization problem in equation 2 is then solved for the two variables $\mathbf{D}$ and $\{\boldsymbol{\alpha}^{(k)}\}_{k=1}^{K}$ in an alternating fashion (Mairal et al., 2009; Elad & Aharon, 2006).

The learnt dictionary atoms form the first layer of our frontend, so that the first layer outputs are the projections of the patches onto the atoms. Specifically, for a given image $\mathbf{X}$, patch $\mathbf{x}_{ij} \in \mathbb{R}^{\bar{n}}$ is extracted based on the (ij)-th block of $\mathbf{X}$; that is, $\mathbf{x}_{ij} = \mathbf{R}_{ij}\mathbf{X}$, and then projected onto dictionary $\mathbf{D}$ in order to obtain projection vector $\tilde{\mathbf{x}}_{ij}$, where $\tilde{\mathbf{x}}_{ij} = \mathbf{D}^T\mathbf{x}_{ij}$.

Since the dictionary is highly overcomplete, the majority of coefficients are relatively small, and a sparse representation of the patch can be obtained from a subset of the coefficients taking large values. Fig. 1a contrasts the typical distribution of coefficients from our dictionary to that for the first layer for a typical CNN (the latter is not amenable to sparse coding).

3.2 COEFFICIENT SELECTION AND ACTIVATION

For each patch, we keep only the T elements of the projection vector with largest absolute values, zeroing out the remaining elements, limiting the dimension of the subspace available to the adversary. The surviving coefficients are denoted by $\hat{\mathbf{x}}_{ij}$. Next, in order to ensure that the adversarial perturbations cannot ride on top of the signal in the surviving coefficients $\hat{\mathbf{x}}_{ij}$, we apply an activation function, zeroing out coefficients smaller than a threshold, and keeping only the sign of coefficients passing the threshold. The specific design we consider targets ℓ^∞ bounded perturbations of size ϵ (but as we shall see, provides robustness against ℓ^2 and ℓ^1 bounded perturbations as well): we set the threshold for atom $\mathbf{d}_l$ to be a multiple of $\epsilon\,\|\mathbf{d}_l\|_1$, the maximum possible contribution of of such a perturbation to the output.

$$\bar{\mathbf{x}}_{ij}(l) = \begin{cases} \operatorname{sign}\left(\hat{\mathbf{x}}_{ij}(l)\right) \|\mathbf{d}_l\|_1 , & \text{if } \frac{|\hat{\mathbf{x}}_{ij}(l)|}{\epsilon\|\mathbf{d}_l\|_1} \geq \beta \\ 0, & \text{otherwise} \end{cases} , \tag{3}$$

for all $l \in \{1, \dots, L\}$, where $\beta > 1$ is a hyperparameter. The scaling of the surviving ± 1 outputs by $\|\mathbf{d}_l\|_1$ allows basis coefficients surviving a larger ℓ^1 norm based threshold to contribute more towards the decoder input, but could be omitted, since the decoder can learn the appropriate weights.

3.3 DECODER

The discretized sparse codes $\bar{\mathbf{x}}_{ij}$ are fed into a CNN decoder with three transposed convolution layers followed by ReLU activations after each layer, which restores the original image size.

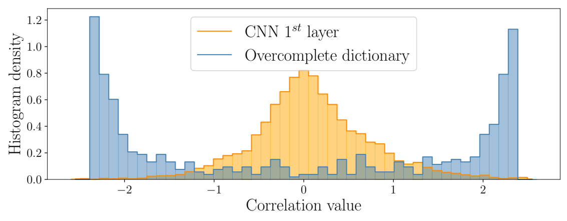
4 EVALUATION

State-of-the-art PGD attacks are based on gradient computation, and an important criticism of prior proposals of preprocessing-based defense is that they were masking/obfuscating/shattering gradients, without adapting the attacks to compensate for this (Athalye et al., 2018). Indeed, many such defenses do fail when attacks are suitably adapted, leading a group of prominent researchers to propose guidelines for evaluating defenses (Carlini et al., 2019). Following such guidelines, we devote substantial effort to devising attacks adapted to our defense as follows:

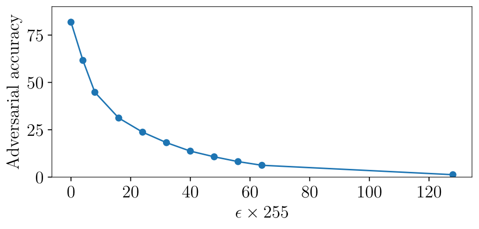
- Consider several different smooth backward pass replacements for the parts of our defense that are non-differentiable:
  - for the activation function, we test two backward passes: identity and a smooth approximation.
  - for top T taking operation, we test identity backward pass as well as propagating gradients through top U coefficients where $U > T$.
- Check that there exist some ϵ values that reduce the adversarial accuracy to 0. (see Fig. 1b)
- Test our defense against attacks with large number of iterations or restarts.
- Test against different threat models such as: ℓ^∞, ℓ^2, and ℓ^1 bounded attacks; decision and query based blackbox attacks that do not rely on gradients; transfer blackbox attacks with many different surrogate models
- To ensure that the gradients are computed appropriately, we test computing smoothed gradients where gradients are averaged over many different points in the neighborhood of the original point, in each step of the attack. However, we observe that this does not significantly increase the attack strength.

5 EXPERIMENTS AND RESULTS

We test our defense against a wide range of attacks and compare the results with benchmark defenses from the literature using the CIFAR-10 dataset (pixels scaled between 0 and 1). The benchmarks



(a)



(b)

Figure 1: (a) Histogram of correlations for a typical patch with atoms of an overcomplete dictionary vs. that of activations through layer 1 filters of a standard classifier CNN. (b) Accuracy vs ϵ plot for our defense ($T = 15$) under ℓ^∞ PGD attack with various ϵ values.

are PGD adversarially trained (Madry et al., 2018) and TRADES (Zhang et al., 2019) defenses for the same classifier architecture under ℓ^∞ bounded attacks. See Appendix A.3 for choice of hyperparameters and training settings for both our defense and benchmarks. Note that the classifier CNN used in our paper is "simpler" ResNet-32 rather than the wide ResNet-32, both of which are utilized in Madry et al. (2018) and other studies in the literature. The choice of the smaller ResNet-32 network makes evaluation of stronger attacks computationally more feasible.

Table 1 compares our defense with the benchmarks from the literature (all columns except the last report classification accuracies). In all whitebox attacks, we use attack settings: number of steps $N_s = 40$, number of random restarts $N_r = 100$ and step size $\delta = \epsilon/8$. For the attacks against our defense, we also use $U = 2T$ and backward pass smoothing mentioned in Section 4 to make it stronger. See Appendix A.2 for discussion of these choices and accuracies for other attack variations. Column 1 reports on clean accuracy. In columns 2 and 3, we report results for the ℓ^∞ bounded whitebox attack with cross entropy loss and Carlini-Wagner loss, respectively. In columns 4 and 5, we report results for ℓ^2 and ℓ^1 bounded attacks, respectively. For the last (sixth) column, we use a decision-based blackbox attack (Brendel et al., 2017) and report the average ℓ^2 norm of perturbations that brings the classification accuracy to $\sim 0.01\%$. This attack does not rely on gradient computation. For a given datapoint, the starting point for the perturbation search in this attack is chosen as the closest (in the ℓ^2 sense) datapoint with a different label. Other parameters for this attack are the default parameters in `foolbox` (Rauber et al., 2020) Python package.

We highlight the following observations from Table 1: (a) the clean accuracy (column 1) is better than that of the benchmarks, (b) the adversarial accuracy (columns 2 and 3) for ℓ^∞ bounded perturbations falls short of the benchmarks, (c) the adversarial accuracy for ℓ^2 (column 4) and ℓ^1 (column 5) bounded perturbations is substantially better than that of the benchmarks, (d) the average ℓ_2 norm (column 6) required to create a classification error with high probability with a non-gradient attack is substantially higher for our defense.

The robustness of our defense to ℓ^p attacks with different p is because the impact of different attacks is similar: they flip ± 1 to 0 and 0 to ± 1 in the sparse code. In contrast, standard adversarial training approaches are optimized for ℓ^p bounded attacks for a specific p, and do not perform as well under other types of attacks.

6 Conclusion

The promising results obtained with our sparse coding frontend open up an exciting new direction for building bottom-up defenses to make neural networks more robust for a wider range of attacks: we can tune our defense to approach the performance of state-of-the-art adversarial training for ℓ^∞ bounded attacks, while also providing robustness against ℓ^2 and ℓ^1 bounded attacks. The ideas that our defense is based on could be applied to not only the first layer, but also to intermediate layers within neural networks. Such an approach could distribute the workload of combatting attacks across layers, and potentially make the networks more robust to a variety of attacks from within, without the need for adversarial training adapted to a specific class of attacks.

Acknowledgment

This work was supported in part by the Army Research Office under grant W911NF-19-1-0053, and by the National Science Foundation under grants CIF-1909320 and CNS-1518812.

	Clean	Whitebox (ℓ^∞, $\epsilon = \frac{8}{255}$)	Whitebox (C&W) (ℓ^∞, $\epsilon = \frac{8}{255}$)	Whitebox (ℓ^2, $\epsilon = 0.6$)	Whitebox (ℓ^1, $\epsilon = 30$)	Decision Boundary BB
Natural	**92.66**	0.00	0.00	0.00	0.00	$\overline{\lvert\lvert \mathbf{e} \rvert\rvert_2} = 0.11$
Adv. Training	79.41	43.27	42.03	52.09	19.98	$\overline{\lvert\lvert \mathbf{e} \rvert\rvert_2} = 0.88$
TRADES	75.17	**45.79**	**42.87**	51.35	21.15	$\overline{\lvert\lvert \mathbf{e} \rvert\rvert_2} = 0.91$
$T = 15$ (Ours)	85.45	37.33	37.35	**60.47**	**47.13**	$\overline{\lvert\lvert \mathbf{e} \rvert\rvert_2} = \mathbf{3.04}$

Table 1: Performance (accuracies and norms) of different defense methods (CIFAR-10)

References

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 387–402. Springer, 2013.

Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv:1712.04248*, 2017.

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM Workshop on Artificial Intelligence and Security*, pp. 3–14, 2017.

Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv:1902.06705*, 2019.

Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems*, pp. 11192–11203, 2019.

Jianbo Chen, Michael I Jordan, and Martin J Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*, pp. 1277–1294. IEEE, 2020.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, pp. 15–26, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450352024. doi: 10.1145/3128572.3140448. URL `https://doi.org/10.1145/3128572.3140448`.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*, pp. 854–863. PMLR, 2017.

Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv:1902.02918*, 2019.

Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 15(12):3736–3745, 2006.

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

Hoki Kim. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*, 2020.

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 (ICLR)*, 2018.

Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pp. 689–696, 2009.

Jonas Rauber, Roland Zimmermann, Matthias Bethge, and Wieland Brendel. Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax. *Journal of Open Source Software*, 5(53):2607, 2020. doi: 10.21105/joss.02607. URL `https://doi.org/10.21105/joss.02607`.

Leslie N Smith. Cyclical learning rates for training neural networks. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

Yuzhe Yang, Guo Zhang, Dina Katabi, and Zhi Xu. ME-Net: Towards effective adversarial robustness with matrix estimation. *arXiv:1905.11971*, 2019.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. *arXiv:1901.08573*, 2019.

A Appendix

A.1 Block Diagram of the Defense

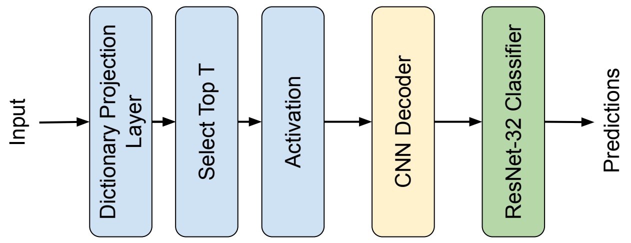


Figure 2: A block diagram of our proposed defense

Input images first go through the convolutional layer with the dictionary atoms being its filters. Then, for each patch representation, Top T coefficients in absolute value are kept while other coefficients are zeroed out. This is followed by the activation function described in Section 3.2. These quantized and highly sparse representations are then fed into a convolutional neural net based decoder which restores the image size. The outputs of the frontend are the inputs of the ResNet-32 based classifier. Training of the decoder and the classifier are done simultaneously after the dictionary is learnt and frozen.

A.2 Determining Attack Parameters

In order to determine the strongest attack settings against our defense, we test it with different attack parameters. In Table 2, we report five whitebox attack settings on our defense for different T values. For these attacks, every differentiable operation is differentiated. For the non-differentiable activation function, we take a smooth backward pass approximation. We also experiment with replacing it with identity in the backward pass but this results in weaker attacks (see A.4). For the baseline attack settings, we choose number of restarts $N_r = 10$, number of steps $N_s = 40$, and step size $\delta = 1/255$. We then change one or two settings at a time and report those in each column. In the third column, we report accuracies when gradients are propagated through the top U coefficients (rather than top T). For the fourth column, we increase N_s to 1000, reduce δ to 0.5 and N_r to 1 in order to keep computational complexity reasonable. For the fifth column, we increase N_r to 100 and keep $N_s = 40$. Finally, for sixth column, we propagate the gradients through the top U coefficients as well as increasing the number of restarts N_r to 100. For all values of T, last two columns' settings result in the strongest attacks. Increasing the number of restarts N_r to 100 especially, had the biggest impact on the accuracies.

By looking at the results in Table 2, we determine to report on $T = 15$ in Table 1, since it represents a good tradeoff between clean and attacked accuracy. After the decision to use $T = 15$ for benchmark comparison in Table 1, we use the strongest attack for this defense setting, which is taking $N_r = 100$ and propagating gradients through the top $U = 2T = 30$ coefficients.

	Clean	Whitebox (Baseline)	Whitebox ($U = 2T$)	Whitebox ($N_s = 1000$)	Whitebox ($N_r = 100$)	Whitebox ($U = 2T$, $N_r = 100$)
$T = 1$	81.81	44.79	48.26	59.91	34.89	37.83
$T = 2$	83.03	47.89	47.81	61.35	37.48	37.92
$T = 5$	83.24	47.11	42.21	60.37	36.66	32.83
$T = 10$	84.71	48.48	44.83	60.06	39.69	36.16
$T = 15$	85.45	46.89	44.55	57.13	38.43	37.33
$T = 20$	85.46	40.96	45.27	51.39	32.63	37.25

Table 2: Accuracies for our defense method under different settings (CIFAR-10, ℓ^∞ $\epsilon = 8/255$)

A.3 Choice of Hyperparameters and Training Settings

Our defense: We evaluate our defense on the CIFAR-10 dataset ($N = 32$), for which there are well-established benchmarks in adversarial ML. In our defense, we use 4×4 patches ($n = 4$) and an overcomplete dictionary with $L = 500$ atoms. The stride $S = 2$, so the encoder output is a $15 \times 15 \times 500$ tensor ($m = 15$, $L = 500$). The regularization parameter in equation 2 is set to $\lambda = 1$, in the upper range of values resulting in convergence. The number of iterations in dictionary learning is chosen as 1000 to ensure convergence. The number of dictionary atoms L is chosen to be 10 times the ambient dimension of patches.

We test our defense for $T = 1, 2, 5, 10, 15,$ and 20 with hyperparameter $\beta = 3$ for the threshold in equation 3. We train the CNN-based decoder in supervised fashion in tandem with the classifier, using the standard cross-entropy loss. We use a cyclic learning rate scheduler (Smith, 2017) with a maximum learning rate of $\eta_{max} = 0.05$ for $T = 1, 2$ and $\eta_{max} = 0.02$ for $T = 5, 10, 15, 20$. In order to provide a consistent evaluation, we employ the ResNet-32 classifier used in Madry et al. (2018) and train it for 70 epochs.

Benchmarks: For a fair comparison, we use the same ResNet-32 classifier architecture for the benchmarks. We train the PGD adversarially trained model from Madry et al. (2018) with the same cyclic learning rate with $\eta_{max} = 0.05$ for 100 epochs. We train the model for TRADES defense with learning rate $\eta = 0.01$ for the first 50 epochs and then with $\eta = 10^{-3}$ for the next 50 epochs. For both PGD adversarially trained model and TRADES, training hyperparameters are $\epsilon = 8/255$, $\delta = 1/255$, $N_S = 10$, $N_R = 1$. Additionally for TRADES $\lambda_{\text{TRADES}} = 1/6$. We also report on naturally trained network (i.e., no defense). This network is also trained for 70 epochs with the same cyclic learning rate with $\eta_{max} = 0.05$.

A.4 Backward Pass Approximation to Activation

We try replacing the activation function with two different functions in the backward pass: identity and a smooth approximation (Fig. 3). We observe that the approximation with steepness$= 4.0$ results in the strongest attacks and use it in the backward pass of all reported attacks.

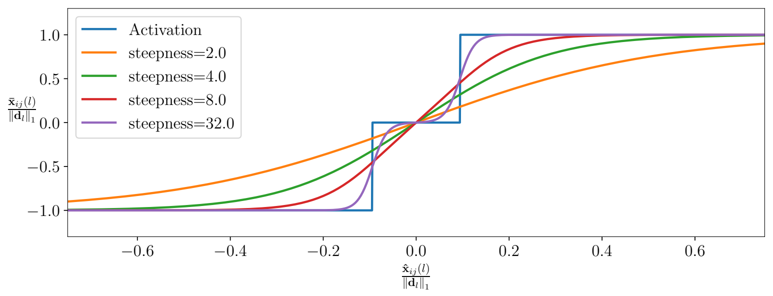


Figure 3: Activation function and its backward pass smooth approximation with varying degrees of smoothness

A.5 Gradient Propagation Through Top U Coefficients

We test different settings for the hyperparameter U such as $U = 2T$, $U = 5T$, $U = 10T$ and find that, in general, $U = 2T$ results in the strongest attacks.

A.6 Gradient Smoothing

We test smoothing the gradients in each step by averaging them over 40 different points in the δ ℓ^∞ neighborhood in each step of the attack. For computational feasibility, we reduce N_s to 20 and N_r to 1. In general, for the models we tried this on, this smoothing operation did not result in significantly stronger attacks. It decreased accuracies less than 0.5% while increasing the computational cost significantly. For this reason, we choose not to use this operation in the attacks reported.

A.7 Results for HopSkipJumpAttack

HopSkipJumpAttack (Chen et al., 2020) is a recently proposed decision-based attack. The results for this attack are reported in Table 3. Similar to the last column of Table 1 the reported results are for the average ℓ^2 norm of the perturbations. We observe that, like the decision boundary attack, our defense requires the highest average ℓ^2 norm perturbations.

	HopSkipJumpAttack
	Average ℓ^2 norm of successful attack images
Natural	$\overline{\|\|\mathbf{e}\|\|_2} = 0.31$
Adv. Training	$\overline{\|\|\mathbf{e}\|\|_2} = 2.08$
TRADES	$\overline{\|\|\mathbf{e}\|\|_2} = 2.00$
$T = 15$ (Ours)	$\overline{\|\|\mathbf{e}\|\|_2} = \mathbf{3.43}$

Table 3: Results for HopSkipJumpAttack attack (CIFAR-10)

A.8 Results for Zeroth Order Optimization (ZOO) Based Attack

ZOO attack (Chen et al., 2017) is a query-based blackbox attack where the gradients of the model with respect to each input pixel are approximated using numerical differentiation and then the attack is computed through stochastic coordinate descent. For computational complexity reasons we evaluate this attack on 100 randomly selected images. These do not include images that are wrongly classified by the corresponding model. The results for this attack are reported in Table 4. Similar to the last column of Table 1 the reported results are for the average ℓ^2 norm of the perturbations. We observe that the ZOO attack is unable to find adversarial examples for our defense. This is due to the attack computing gradients by changing one pixel at a time by $\pm\Delta x$. Since our frontend is very insensitive to single pixel changes, the gradients cannot be calculated using such numerical differentiation techniques. We acknowledge that while this observation alone doesn't show that our defense is secure, it does mean that this particular type of attack is not applicable to our defense.

	ZOO Attack	
	Attack success rate (%)	Average ℓ^2 norm of successful attack images
Natural	100	$\overline{\|\|\mathbf{e}\|\|_2} = 0.13$
Adv. Training	100	$\overline{\|\|\mathbf{e}\|\|_2} = 0.89$
TRADES	100	$\overline{\|\|\mathbf{e}\|\|_2} = 0.93$
$T = 15$ (Ours)	7	$\overline{\|\|\mathbf{e}\|\|_2} = 0.04$

Table 4: Results for ZOO attack (CIFAR-10)

A.9 Effect of Attack Step Size

Nominally, we report results for when the stepsize $\delta = \epsilon/8$. For the ℓ^∞ attack, when we use $N_s = 1000$, we decrease δ to $0.5/255$. We also test for $\delta = 2/255$ but this results in higher accuracies therefore is omitted in the tables. For the ℓ^1 and ℓ^2 bounded attacks, we try $\delta = \epsilon/20$ but this too results in weaker attacks.

A.10 Validation of Attack Code

To generate all attacks except for the decision-based blackbox attacks, we use our own attack library and validate our results with the `foolbox` (Rauber et al., 2020) and `torchattacks` (Kim, 2020) Python packages. To generate the decision based blackbox attacks, we use the `foolbox` package.