Adversarial Neural Pruning with Latent Vulnerability Suppression

Divyam Madaan [1] **Jinwoo Shin** [2 3] **Sung Ju Hwang** [1 3 4]

Abstract

Despite the remarkable performance of deep neural networks on various computer vision tasks, they are known to be susceptible to adversarial perturbations, which makes it challenging to deploy them in real-world safety-critical applications. In this paper, we conjecture that the leading cause of adversarial vulnerability is the distortion in the latent feature space, and provide methods to suppress them effectively. Explicitly, we define *vulnerability* for each latent feature and then propose a new loss for adversarial learning, *Vulnerability Suppression (VS)* loss, that aims to minimize the feature-level vulnerability during training. We further propose a Bayesian framework to prune features with high vulnerability to reduce both vulnerability and loss on adversarial samples. We validate our *Adversarial Neural Pruning with Vulnerability Suppression (ANP-VS)* method on multiple benchmark datasets, on which it not only obtains state-of-the-art adversarial robustness but also improves the performance on clean examples, using only a fraction of the parameters used by the full network. Further qualitative analysis suggests that the improvements come from the suppression of feature-level vulnerability.

1. Introduction

In the last many years, deep neural networks (DNNs) have achieved impressive results on various computer vision tasks, e.g., image classification (Krizhevsky et al., 2012), face/object recognition (He et al., 2015; Deng et al., 2019), and semantic segmentation (He et al., 2017). The groundbreaking success of DNNs has motivated their use in safety-critical environments such as medical imaging (Esteva et al., 2017) and autonomous driving (Bojarski et al., 2016). However, DNNs are extremely brittle to carefully crafted imperceptible adversarial perturbations intentionally optimized to cause miss-prediction (Szegedy et al., 2013; Goodfellow et al., 2014).

While the field has primarily focused on the development of new attacks and defenses, a 'cat-and-mouse' game between attacker and defender has arisen. There has been a long list of proposed defenses to mitigate the effect of adversarial examples, e.g., defenses (Papernot et al., 2016; Xu et al., 2017b; Buckman et al., 2018; Dhillon et al., 2018), followed by successful attacks (Carlini & Wagner, 2016; Athalye et al., 2018; Uesato et al., 2018). This shows that any defense mechanism that once looks successful could be circumvented with the invention of new attacks. In this paper, we tackle the problem by identifying a more fundamental cause of the adversarial vulnerability of DNNs.

What makes DNNs vulnerable to adversarial attacks? Our intuition is that the adversarial vulnerability of a DNN comes from the distortion in the *latent feature* space incurred by the adversarial perturbation. If a perturbation at the input level is suppressed successfully at any layers of the DNN, then it would not cause miss-prediction. However, not all latent features will contribute equally to the distortion; some features may have more substantial distortion, by amplifying the perturbations at the input level, while others will remain relatively static. Under this motivation, we first formally define the vulnerability of the latent features and show that sparse networks can have a much smaller degree of network-level vulnerability in Figure 2(a). Then, we propose an effective way to suppress vulnerability by pruning the latent features with high vulnerability. We refer to this defense mechanism as *Adversarial Neural Pruning (ANP)*. Further, we propose a novel loss, which we refer to as *Vulnerability Suppression (VS)* loss, that directly suppresses the vulnerability in the feature space by minimizing the feature-level vulnerability. To this end, we propose a novel defense mechanism coined, *Adversarial Neural Pruning with Vulnerability Suppression (ANP-VS)*, that learns pruning masks for the features in a Bayesian framework to minimize both the adversarial loss and the feature-level vulnerability, as illustrated in Figure 1. It effectively suppresses the distortion in the latent feature space with almost no extra cost relative to adversarial training and yields light-weighted networks that are more robust to adversarial perturbations.

[1]School of Computing, KAIST, South Korea [2]School of Electrical Engineering, KAIST, South Korea [3]Graduate School of AI, KAIST, South Korea [4]AITRICS, South Korea. Correspondence to: Divyam Madaan <dmadaan@kaist.ac.kr>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

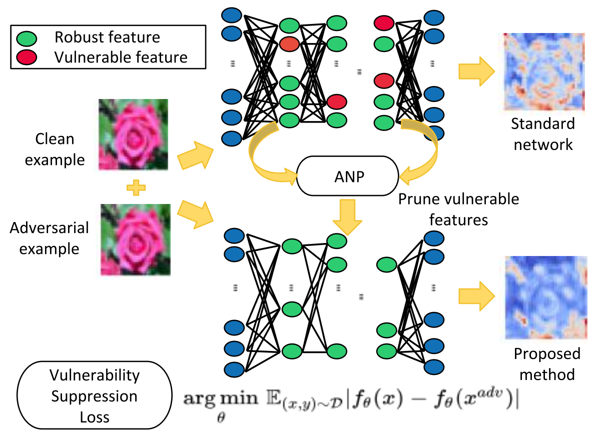

Robust feature
Vulnerable feature
Clean example
Adversarial example
ANP
Prune vulnerable features
Standard network
Proposed method
Vulnerability Suppression Loss
$\arg\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}|f_\theta(x) - f_\theta(x^{adv})|$

Figure 1. **Concept:** We hypothesize that the distortion in the latent feature space is the leading cause of the adversarial vulnerability of deep neural networks and introduce a novel *Vulnerability Suppression (VS)* loss that explicitly aims to minimize feature-level distortions. Further, we prune the features with significant distortions by learning pruning masks with *Adversarial Neural Pruning (ANP)* that minimizes the adversarial loss.

In summary, the contributions of this paper are as follows:

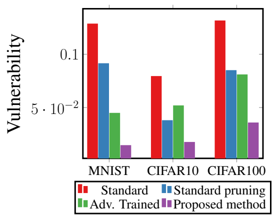
- We hypothesize that the distortion in the latent features is the leading cause of DNN's susceptibility to adversarial attacks and formally describe the concepts of the vulnerability of latent-features based on the expectation of the distortion of latent-features with respect to input perturbations.
- Based on this finding, we introduce a novel defense mechanism, *Adversarial Neural Pruning with Vulnerability Suppression (ANP-VS)*, to mitigate the feature-level vulnerability. The resulting framework learns a Bayesian pruning (dropout) mask to prune out the vulnerable features while preserving the robust ones by minimizing the adversarial and *Vulnerability Suppression (VS)* loss.
- We experimentally validate our proposed method on MNIST, CIFAR-10, and CIFAR-100 datasets, on which it achieves state-of-the-art robustness with a substantial reduction in memory and computation, with qualitative analysis which suggests that the improvement on robustness comes from its suppression of feature-level vulnerability.

While our primary focus is on improving the robustness of DNNs, we also found that ANP-VS achieves higher accuracy for clean/non-adversarial inputs, compared to baseline adversarial training methods (see the results of CIFAR datasets in Table 1). This is another essential benefit of ANP-VS as it has been well known that adversarial training tends to hurt the clean accuracy (Schmidt et al., 2018; Tsipras et al., 2019; Zhang et al., 2019).

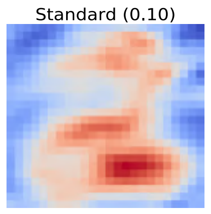
2. Related work

Adversarial robustness. Since the literature on the adversarial robustness of neural networks is vast, we only discuss some of the most relevant studies. A large number of defenses (Papernot et al., 2016; Xu et al., 2017a; Buckman et al., 2018; Dhillon et al., 2018; Song et al., 2018) have been proposed and consequently broken by more sophisticated attack methods (Carlini & Wagner, 2016; Athalye et al., 2018; Uesato et al., 2018). Adversarial training based defenses (Madry et al., 2018; Liu et al., 2019; Hendrycks et al., 2019; Zhang et al., 2019) are widely considered to be the most effective since they largely avoid the obfuscated gradients problem (Athalye et al., 2018). There has also been previous work that studied robust and vulnerable features at the input level. Garg et al. (2018) established a relation between adversarially robust features and the spectral property of the geometry of the dataset and Gao et al. (2017) proposed to remove unnecessary features to get robustness. Recently, Ilyas et al. (2019) disentangle robust and non-robust features in the input-space. Our work is different from these existing works in that we consider and define the vulnerability at the latent feature level, which is more directly related to the model prediction.

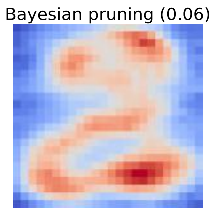
Sparsification methods. Sparsification of neural networks is becoming increasingly important with the increased deployments of deep network models to resource-limited devices. The most straightforward way to sparsify neural networks is by removing weights with small magnitude (Strm, 1997; Collins & Kohli, 2014); however, such heuristics-based pruning often degenerates accuracy, and Han et al. (2015) proposed an iterative retraining and pruning approach to recover from the damage from pruning. Using sparsity-inducing regularization (e.g. ℓ_1) is another popular approach for network sparsification. However elementwise sparsity does not yield practical speed-ups and Wen et al. (2016) proposed to use group sparsity to drop a neuron or a filter as a whole, that will reduce the actual network size. Molchanov et al. (2017) proposed to learn the individual dropout rates per weight with sparsity-inducing priors to completely drop out unnecessary weights, and Neklyudov et al. (2017) proposed to exploit structured sparsity by learning masks for each neuron or filter. Lee et al. (2018) proposed a variational dropout whose dropout probabilities are drawn from sparsity-inducing beta-Bernoulli prior. Information-theoretic approaches have been also shown to be effective, such as Dai et al. (2018) which minimizes the information theoretic bound to reduce the redundancy between layers.



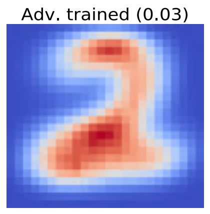
(a) Mean distortion



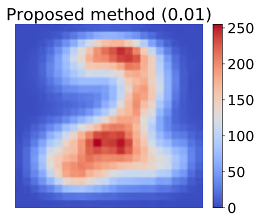
(b) Standard



(c) Bayesian Pruning



(d) Adv. Training



(e) ANP-VS (Ours)

Figure 2. (a) Mean distortion (average perturbation in latent features across all layers) for MNIST on Lenet-5-Caffe, CIFAR-10 and CIFAR-100 on VGG-16. Our method yields a network with minimum vulnerability (distortion) compared to all the other networks; we provide the formal definition of vulnerability in Section 3. Visualization of the vulnerability for the input layer for Lenet-5-Caffe on MNIST for various methods where the vulnerability of the input layer is reported in the bracket (the smaller is, the better): (b) Standard trained model, (c) Standard Bayesian pruning, (d) Adversarial trained network and (e) ANP-VS. The standard network (a) has the maximum distortion, which is comparatively reduced in adversarial training (d) and further suppressed by ANP-VS (e).

Robustness and sparsity. The sparsity and robustness have been explored together in various recent works. Guo et al. (2018) analyzes sparsity and robustness from a theoretical and experimental perspective and demonstrate that appropriately higher sparsity leads to a more robust model and Ye et al. (2018) experimentally discuss how pruning shall effect robustness with a similar conclusion. In contrary, Wang et al. (2018) derived opposite conclusions showing that robustness decreases with increase in sparsity. Ye & Xu (2019) proposed concurrent adversarial training and weight pruning to enable model compression while preserving robustness. However, ANP-VS is fundamentally different from all these methods; we sparsify networks while explicitly learning the pruning (dropout) mask to minimize the loss on adversarial examples.

3. Robustness of deep representations

We first briefly introduce some notations and the concept of vulnerability in the deep latent representation space. We represent a L-layer neural network by a function $f:\ \boldsymbol{X}\rightarrow\boldsymbol{Y}$ with dataset denoted by $\mathcal{D}=\{\boldsymbol{X},\boldsymbol{Y}\}$. Specifically, we denote any sample instance by $\boldsymbol{x}\in\boldsymbol{X}$, and it's corresponding label by $\boldsymbol{y}$. The output vector $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ can then be represented by $f_{\boldsymbol{\theta}}(\boldsymbol{x})=f_{L-1}(f_{L-2}(\cdots(f_1(\boldsymbol{x}))))$ where, $\boldsymbol{\theta}=\{\boldsymbol{W}_1,\ldots,\boldsymbol{W}_{L-1},\boldsymbol{b}_1,\ldots,\boldsymbol{b}_{L-1}\}$.

Let $\boldsymbol{z}_l$ denote the latent-feature vector for the l-th layer with rectified linear unit (ReLU) as the activation function, then $f_l(\cdot)$ can be defined as:

$$\boldsymbol{z}_{l+1}=f_l(\boldsymbol{z}_l)=\max\{\boldsymbol{W}_l\boldsymbol{z}_l+\boldsymbol{b}_l,0\},\quad\forall l\in\{1,2,\ldots,L-2\}$$

where $\boldsymbol{W}_l$ and $\boldsymbol{b}_l$ denote the weight and bias vector respectively. Let $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ denote clean and adversarial data points, respectively ($\tilde{\boldsymbol{x}}=\boldsymbol{x}+\delta$) for any $\boldsymbol{x}\in\boldsymbol{X}$ with ℓ_p-ball $\mathcal{B}(\boldsymbol{x},\varepsilon)$ around $\boldsymbol{x}$: $\{\tilde{\boldsymbol{x}}\in\boldsymbol{X}:\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|\leq\varepsilon\}$, $\boldsymbol{z}_l$ and $\tilde{\boldsymbol{z}}_l$ as their corresponding latent-feature map vectors for the l-th layer of the network.

Vulnerability of a latent-feature. The vulnerability of a k-th latent-feature for l-th layer can be measured by the distortion in that feature in the presence of an adversary. The vulnerability of a latent feature could then be defined as the expectation of the Manhattan distance between the feature value for a clean example ($\boldsymbol{z}_{lk}$) and its adversarial example ($\tilde{\boldsymbol{z}}_{lk}$). This could be formally defined as follows:

$$\mathrm{v}(\boldsymbol{z}_{lk},\tilde{\boldsymbol{z}}_{lk})=\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}||\boldsymbol{z}_{lk}-\tilde{\boldsymbol{z}}_{lk}||\tag{1}$$

Vulnerability of a network. We can measure the vulnerability of an entire network $f_{\boldsymbol{\theta}}(\boldsymbol{X})$ by computing the sum of the vulnerability of all the latent features vectors of the network before the logit layer, then $\mathrm{V}(f_{\boldsymbol{\theta}}(\boldsymbol{X}),f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{X}}))$ can be defined as:

$$\begin{aligned}\overline{\mathrm{v}_l}&=\frac{1}{N_l}\sum_{k=1}^{k=N_l}\mathrm{v}(\boldsymbol{z}_{lk},\tilde{\boldsymbol{z}}_{lk}),\\ \mathrm{V}(f_{\boldsymbol{\theta}}(\boldsymbol{X}),f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{X}}))&=\frac{1}{L-2}\sum_{l=1}^{l=L-2}\overline{\mathrm{v}_l}\end{aligned}\tag{2}$$

where $\overline{\mathrm{v}_l}$ represents the vulnerability of the layer l composed of N_l latent-features. Figure 2(a) shows the vulnerability of different networks across various datasets. It can be observed that although adversarial training suppresses the vulnerability at the input level, the latent feature space is still vulnerable to adversarial perturbation and that our proposed method achieves the minimum distortion.

Adversarial training. Adversarial Training (Goodfellow et al., 2014; Kurakin et al., 2016) was proposed as a data augmentation method to train the network on the mixture of clean and adversarial examples until the loss converges. Madry et al. (2018) incorporated the adversarial search inside the training process by solving the following non-convex outer minimization problem and a non-concave inner maximization problem:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{B}(\boldsymbol{x},\varepsilon)} \mathcal{L}(\boldsymbol{\theta},\tilde{\boldsymbol{x}},\boldsymbol{y})\right] \tag{3}$$

Figure 2 shows that adversarial training can distinguish between robust and vulnerable features. While the standard training results in obtaining vulnerable features, adversarial training reduces the vulnerability of the latent-features and selects the robust features which are necessarily required to attain adversarial robustness.

Relationship between robustness and sparsity. Guo et al. (2018); Ye et al. (2018) shows that sparsifying networks leads to more robust networks, which is evident by our definitions: sparsity suppresses vulnerability to 0 and thus reduces the network vulnerability. Ye & Xu (2019) uses a pre-trained adversarial defense model and investigated different pruning schemes for robustness. However, the network still does not take into account the robustness of a latent-feature. To address these limitations, we introduce *ANP-VS* in the next section, which is a novel method to prune and suppress these vulnerable features and only requires a standard network. In the experiments section, we show that our method significantly outperforms their method in adversarial robustness and computational efficiency.

4. Adversarial neural pruning with vulnerability suppression

In this section, we propose our method to reduce the vulnerability in the latent space. Let $\mathcal{L}(\boldsymbol{\theta}\odot\boldsymbol{M},\boldsymbol{x},\boldsymbol{y})$ be the loss function at data point $\boldsymbol{x}$ with class $\boldsymbol{y}$ for any $\boldsymbol{x}\in\boldsymbol{X}$ for the model with parameters $\boldsymbol{\theta}$ and mask parameters $\boldsymbol{M}$, we use Projected Gradient Descent (PGD) (Madry et al., 2018) to generate the adversarial examples:

$$\tilde{\boldsymbol{x}}^{k+1} = \prod_{\mathcal{B}(\boldsymbol{x},\varepsilon)} \left(\tilde{\boldsymbol{x}}^{k} + \alpha\cdot\operatorname{sgn}(\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{\theta}\odot\boldsymbol{M},\tilde{\boldsymbol{x}}^{k},\boldsymbol{y}))\right) \tag{4}$$

where α is the step size, $\operatorname{sgn}(\cdot)$ returns the sign of the vector, $\odot$ is the hadamard product, $\prod(\cdot)$ is the projection operator on ℓ_∞ norm-ball $\mathcal{B}(\boldsymbol{x},\varepsilon)$ around $\boldsymbol{x}$ with radius ε for each example, and $\boldsymbol{x}^{k+1}$ denotes the adversarial example at the k-th PGD step. Specifically, PGD perturbs the clean example $\boldsymbol{x}$ for K number of steps, and projects the adversarial example $\tilde{x}^{k+1}$ onto the norm-ball of $\boldsymbol{x}$, if it goes beyond $\mathcal{B}(\boldsymbol{x},\varepsilon)$ after each step of perturbation.

In order to minimize the vulnerability of the network, we first propose *Adversarial Neural Pruning (ANP)*. The basic idea of ANP is to achieve robustness while suppressing the distortion, by explicitly pruning out the latent features with high distortion. Specifically, ANP learns pruning masks for the features in a Bayesian framework to minimize the following adversarial loss:

```
Algorithm 1 Adversarial training by ANP-VS
input Dataset $\mathcal{D}$, training iterations $T$, trained model $f_{\boldsymbol{\theta}}$
    with parameters $\boldsymbol{\theta}$, mask parameters $\boldsymbol{M}$, batch size $n$,
    PGD iterations K, and PGD step size $\alpha$ for some norm-
    ball $\mathcal{B}$.
output Pruned state of network $f_{\boldsymbol{\theta}}$
 1: for $t = \{1,\ldots,T\}$ do
 2:     Sample mini-batch $B = \{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_m\} \subset \mathcal{D}$
 3:     for $i = \{1,\ldots,n\}$ do
 4:         // Run PGD adversary
 5:         for $k = \{1,\ldots,K\}$ do
 6:             $\tilde{\boldsymbol{x}}_i^{k+1} = \prod_{\mathcal{B}(\boldsymbol{x}_i,\varepsilon)} (\tilde{\boldsymbol{x}}_i^k + \alpha\,\operatorname{sgn}(\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{\theta}\odot\boldsymbol{M},\tilde{\boldsymbol{x}}_i^k,\boldsymbol{y}_i)))$
 7:         end for
 8:         Optimize $\boldsymbol{\theta}$ by Equation 6 and $\boldsymbol{M}$ by Equation 5
            using gradient descent.
 9:     end for
10: end for
```

$$\min_{\boldsymbol{M}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}\left\{\max_{\delta\in\mathcal{B}(\boldsymbol{x},\varepsilon)} \mathcal{L}(\boldsymbol{\theta}\odot\boldsymbol{M},\tilde{\boldsymbol{x}},\boldsymbol{y})\right\} \tag{5}$$

We further combine our proposed pruning scheme with our novel *vulnerability suppression loss (VS)* that minimizes the network vulnerability and further improves the robustness of the model. Let $J(\boldsymbol{\theta}\odot\boldsymbol{M},\boldsymbol{x},\boldsymbol{y})$ be the adversarial training loss on a batch of samples (comprised of the cross-entropy loss on the clean and adversarial samples, plus any weight decay, etc.). In particular, Adversarial Neural Pruning with Vulnerability Suppression (ANP-VS) optimizes the network parameters $\boldsymbol{\theta}$ using the following loss:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}\left\{\underbrace{J(\boldsymbol{\theta}\odot\boldsymbol{M},\boldsymbol{x},\boldsymbol{y})}_{\text{classification loss}} + \underbrace{\lambda\cdot\mathrm{V}(f_{\boldsymbol{\theta}}(\boldsymbol{x}),f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}))}_{\text{vulnerability suppression loss}}\right\} \tag{6}$$

where λ is the hyper-parameter determining the strength of the vulnerability suppression loss. The overall training algorithm is displayed in Algorithm 1.

Intution behind ANP-VS. ANP in Equation (5) encourages the pruning mask to be optimized by minimizing the adversarial loss and preserving the robustness of the model. The VS loss encourages the suppression of the distortion of the latent features in the presence of adversarial perturbations via minimizing the difference between latent-features of clean examples and those of adversarial examples. We empirically found that ANP-VS also has an effect of increasing the smoothness of the model's output and its loss surface, which is conceptually consistent with the indispensable properties of robust models (Cisse et al., 2017; Tsipras et al., 2019) (see Figure 5).

Adversarial beta-Bernoulli dropout. Beta-Bernoulli Dropout (Lee et al., 2018) learns to set the dropout rate by generating the dropout mask from sparsity-inducing beta-Bernoulli prior to each neuron. Let $\boldsymbol{\theta} \in \mathbb{R}^{K \times L \times M}$ be a parameter tensor of neural network layer with K channels and $\boldsymbol{M} = \{\boldsymbol{m}_1, \dots, \boldsymbol{m}_n\}$ for any $\boldsymbol{m} \in \{0,1\}^K$ be the binary mask sampled from the finite-dimensional beta-Bernoulli prior to be applied for the n-th observation $\boldsymbol{x}_n$.

The goal of the variational inference is to compute the posterior distribution $p(\boldsymbol{\theta}, \boldsymbol{M}, \boldsymbol{\pi}|\mathcal{D})$ and we approximate this posterior using an approximate variational distribution $q(\boldsymbol{\theta}, \boldsymbol{M}, \boldsymbol{\pi}|\tilde{\boldsymbol{X}})$ of known parametric form. For $\boldsymbol{\pi}$, we use the Kumaraswamy distribution (Kumaraswamy, 1980) with parameters a and b following following Lee et al. (2018); Nalisnick & Smyth (2016), and $\boldsymbol{m}_k$ is sampled by reparametrization with continuous relaxation:

$$\begin{aligned} q(\pi_k; a_k, b_k) &= a_k b_k \pi_k^{a_k - 1}(1 - \pi_k^{a_k})^{b_k - 1} \\ \boldsymbol{m}_k &= \mathrm{sgm}\left(\frac{1}{\tau}\left(\log \frac{\pi_k}{1-\pi_k} + \log \frac{u}{1-u}\right)\right) \end{aligned} \quad (7)$$

where $\mathrm{sgm}(x) = \frac{1}{1+e^{-x}}$, $u \sim \mathrm{unif}[0,1]$, and τ is a temperature continuous relaxation. The KL-divergence between the prior and variational distribution can then be computed in a closed form (Nalisnick & Smyth, 2016):

$$\begin{aligned} &D_{KL}[q(\boldsymbol{M}, \boldsymbol{\pi}) \| p(\boldsymbol{M}, \boldsymbol{\pi})] \\ &= \sum_{k=1}^{K} \left\{ \frac{a_k - \alpha/K}{a_k}\left(-\gamma - \Psi(b_k) - \frac{1}{b_k}\right) \right. \\ &\qquad \left. + \log \frac{a_k b_k}{\alpha/K} - \frac{b_k - 1}{b_k} \right\} \end{aligned} \quad (8)$$

where γ is Euler-Mascheroni constant, and $\Psi(.)$ is digamma function. Using the Stochastic Gradient Variational Bayes (SGVB) framework (Kingma et al., 2015), we can optimize the variational parameters and get the final loss as follows:

$$\begin{aligned} \min_{\boldsymbol{M}} \; &\left\{ \sum_{n=1}^{N} \mathbb{E}_q[\log p(\boldsymbol{y}_n | f(\tilde{\boldsymbol{x}}_n; \boldsymbol{\theta} \odot \boldsymbol{M}))] \right. \\ &\qquad \left. - \beta \cdot D_{KL}[q(\boldsymbol{M}; \boldsymbol{\pi}) || p(\boldsymbol{M}|\boldsymbol{\pi})] \right\} \end{aligned} \quad (9)$$

where β is the trade-off parameter between network robustness and pruned network size to individually tailor the degree of compression across each layer of the network. The first term in the loss measures the log-likelihood of the adversarial samples w.r.t. $q(\boldsymbol{M}; \boldsymbol{\pi})$ and the second term regularizes $q(\boldsymbol{M}; \boldsymbol{\pi})$ so it doesn't deviate from the prior distribution. We refer ANP-VS to Adversarial Beta Bernoulli dropout with VS for the rest of our paper. We further extend ANP-VS to the Variational information bottleneck (Dai et al., 2018) in the supplementary material.

5. Experiments

5.1. Experimental setup

Baselines and our model. We first introduce various baselines and our model. We compare against following adversarial robustness and compression baselines:

1. **Standard.** Base convolution neural network.
2. **Bayesian Pruning (BP).** Base network with Beta-Bernoulli dropout (Lee et al., 2018).
3. **Adversarial Training (AT).** Adversarial trained network (Madry et al., 2018).
4. **Adversarial Bayesian Neural Network (AT BNN).** Adversarial Bayesian trained network (Liu et al., 2019).
5. **Pre-trained Adversarial Training (Pretrained AT).** Adversarial training on a pre-trained base model (Hendrycks et al., 2019).
6. **Alternating Direction Method of Multipliers (ADMM).** Concurrent weight pruning and adversarial training (Ye & Xu, 2019).
7. **Theoretically Principled Trade-off between Robustness and Accuracy (TRADES).** Explicit trade off between natural and robust generalization (Zhang et al., 2019).
8. **Adversarial Neural Pruning with vulnerability suppression (ANP-VS).** Adversarial neural pruning regularized with vulnerability suppression loss.

Datasets. We validate our method on following benchmark datasets for adversarial robustness:

1. **MNIST.** This dataset (LeCun, 1998) contains 60,000 grey scale images of handwritten digits sized 28 × 28, where there are 5,000 training instances and 1,000 test instances per class. As for the base network, we use LeNet 5-Caffe [1] for this dataset.
2. **CIFAR-10.** This dataset (Krizhevsky, 2012) consists of 60,000 images sized 32 × 32, from ten animal and vehicle classes. For each class, there are 5,000 images for training and 1,000 images for test. We use VGG-16 (Simonyan & Zisserman, 2014) for this dataset with 13 convolutional and two fully connected layers with pre-activation batch normalization and Binary Dropout.
3. **CIFAR-100.** This dataset (Krizhevsky, 2012) also consists of 60,000 images of 32 × 32 pixels as in CIFAR-10 but has 100 generic object classes instead of 10. Each class has 500 images for training and 100 images for test. We use VGG-16 (Simonyan & Zisserman, 2014) similar to CIFAR-10 dataset as the base network for this dataset.

[1]https://github.com/BVLC/caffe/tree/master/examples/mnist

Table 1. Robustness and compression performance for MNIST on Lenet-5-Caffe, CIFAR-10 and CIFAR-100 on VGG-16 architecture under ℓ_∞-PGD attack. All the values are measured by computing mean and standard deviation across 5 trials upon randomly chosen seeds. The best results over adversarial baselines are highlighted in bold. ↑ (↓) indicates that the higher (lower) number is the better.

			Adversarial accuracy (↑)		Vulnerability (↓)		Computational efficiency		
	Model	Clean accuracy (↑)	White box attack	Black box attack	White box attack	Black box attack	Memory (↓)	xFLOPS (↑)	Sparsity (↑)
MNIST	Standard	99.29±0.02	0.00±0.0	8.02±0.9	0.129±0.001	0.113±0.000	100.0±0.00	1.00±0.00	0.00±0.00
	BP	99.34±0.05	0.00±0.0	12.99±0.5	0.091±0.001	0.078±0.001	4.14±0.29	9.68±0.36	83.48±0.54
	AT	99.14±0.02	88.03±0.7	94.18±0.8	0.045±0.001	0.040±0.000	100.0±0.00	1.00±0.00	0.00±0.00
	AT BNN	99.16±0.05	88.44±0.4	94.87±0.2	0.364±0.023	0.199±0.031	200.0±0.00	0.50±0.00	0.00±0.00
	Pretrained AT	**99.18±0.06**	88.26±0.6	94.49±0.7	0.412±0.035	0.381±0.029	100.0±0.00	1.00±0.00	0.00±0.00
	ADMM	99.01±0.02	88.47±0.4	94.61±0.7	0.041±0.002	0.038±0.001	100.0±0.00	1.00±0.00	80.00±0.00
	TRADES	99.07±0.04	89.67±0.4	95.04±0.6	0.037±0.001	0.033±0.001	100.0±0.00	1.00±0.00	0.00±0.00
	ANP-VS (ours)	99.05±0.08	**91.31±0.9**	**95.43±0.8**	**0.017±0.001**	**0.015±0.001**	**6.81±0.35**	**10.57±1.15**	**84.16±0.36**
CIFAR-10	Standard	92.76±0.1	13.79±0.8	41.65±0.9	0.077±0.001	0.065±0.001	100.0±0.00	1.00±0.00	0.00±0.00
	BP	92.91±0.1	14.30±0.5	42.88±1.3	0.037±0.001	0.033±0.001	12.41±0.14	2.34±0.0.03	75.92±0.13
	AT	87.50±0.5	49.85±0.9	63.70±0.6	0.050±0.002	0.047±0.001	100.0±0.00	1.00±0.00	0.00±0.00
	AT BNN	86.69±0.5	51.87±0.9	64.92±0.9	0.267±0.013	0.238±0.011	200.0±0.00	0.50±0.00	0.00±0.00
	Pretrained AT	87.50±0.4	52.25±0.7	66.10±0.8	0.041±0.002	0.036±0.001	100.0±0.00	1.00±0.00	0.00±0.00
	ADMM	78.15±0.7	47.37±0.6	62.15±0.8	0.034±0.002	0.030±0.002	100.0±0.00	1.00±0.00	75.00±0.00
	TRADES	80.33±0.5	52.08±0.7	64.80±0.5	0.045±0.001	0.042±0.005	100.0±0.00	1.00±0.00	0.00±0.00
	ANP-VS (ours)	**88.18±0.5**	**56.21±0.1**	**71.44±0.6**	**0.019±0.000**	**0.016±0.000**	**12.27±0.18**	**2.41±0.04**	**76.53±0.16**
CIFAR-100	Standard	67.44±0.7	2.81±0.2	14.94±0.8	0.143±0.007	0.119±0.005	100.0±0.00	1.00±0.00	0.00±0.00
	BP	69.40±0.7	3.12±0.1	16.39±0.2	0.067±0.001	0.059±0.001	18.59±0.56	1.95±0.04	63.48±0.88
	AT	57.79±0.8	19.07±0.8	32.47±1.4	0.079±0.003	0.071±0.003	100.0±0.00	1.00±0.00	0.00±0.00
	AT BNN	53.75±0.7	19.40±0.6	30.38±0.2	0.446±0.029	0.385±0.051	200.0±0.00	0.50±0.00	0.00±0.00
	Pretrained AT	57.14±0.9	19.86±0.6	35.42±0.4	0.071±0.001	0.065±0.002	100.0±0.00	1.00±0.00	0.00±0.00
	ADMM	52.52±0.5	19.65±0.5	31.30±0.3	0.060±0.001	0.056±0.001	100.0±0.00	1.00±0.00	65.00±0.00
	TRADES	56.70±0.7	21.21±0.3	32.81±0.6	0.065±0.003	0.060±0.003	100.0±0.00	1.00±0.00	0.00±0.00
	ANP-VS (ours)	**59.15±1.2**	**22.35±0.6**	**37.01±1.1**	**0.035±0.001**	**0.030±0.003**	**16.74±0.52**	**2.02±0.05**	**66.80±0.75**

Evaluation setup. We validate our model with three metrics for computational efficiency: i) *Memory footprint* (Memory) - The ratio of space for storing hidden feature maps in pruned model versus original model. ii) *Floating point operations* (xFLOPs) - The ratio of the number of floating-point operations for the original model versus pruned model. iii) *Model size* (Sparsity) - The ratio of the number of zero units in the original model versus the pruned model. We report the clean, adversarial accuracy and vulnerability (Equation (2)) for ℓ_∞ white box and black box attack (Papernot et al., 2017). For generating black-box adversarial examples, we used AT and standard network for adversarial training and standard methods respectively. We utilize the clean and adversarial examples for adversarial training methods, all our results are measured by computing mean and standard deviation across 5 trials with random seeds. We list the hyper-parameters in the supplementary material, and the code is available online [2].

[2] https://github.com/divyam3897/ANP_VS

5.2. Comparison of robustness and generalization

Evaluation on MNIST. For MNIST, we consider a Lenet-5-Caffe model with a perturbation radius of $\varepsilon = 0.3$, perturbation per step of 0.01, 20 PGD steps for training, and 40 PGD steps with random restarts for evaluating the trained model. Our pretrained standard Lenet 5-Caffe baseline model reaches over 99.29% accuracy after 200 epochs averaged across five runs. The results in Table 1 show that ANP-VS significantly improves the adversarial accuracy and vulnerability of the network over all the competing baselines. Namely, with the robust-features, ANP-VS achieves $\sim 3\%$ improvement in adversarial accuracy and $\sim 65\%$ reduction in the vulnerability against white-box and black-box attacks over the state-of-the-art adversarial training frameworks. In addition, it achieves $\sim 93\%$ reduction in memory footprint with significant speedup over standard adversarial training baselines. The results indicate that ANP-VS better suppresses the vulnerability of the network, which we can directly attribute to increased adversarial robustness.

Table 2. Adversarial accuracy of CIFAR-10 and CIFAR-100 for VGG-16 architecture under ℓ_∞-PGD white box attack for various ε values and PGD iterations with the perturbation per step of 0.007. The best results are highlighted in bold.

		Epsilon (ε)				# Iterations			
		0.01	0.03	0.05	0.07	100	200	500	1000
CIFAR-10	Standard	40.54$\pm$0.9	13.79$\pm$0.8	4.16$\pm$0.5	1.46$\pm$0.1	11.76$\pm$0.4	11.73$\pm$0.7	13.71$\pm$0.6	13.68$\pm$0.7
	BP	41.67$\pm$0.6	14.30$\pm$0.5	4.21$\pm$0.4	1.50$\pm$0.1	14.24$\pm$0.5	14.18$\pm$0.3	14.14$\pm$0.2	14.16$\pm$0.3
	AT	65.86$\pm$0.8	49.85$\pm$0.9	34.54$\pm$0.8	22.69$\pm$0.7	49.74$\pm$0.9	49.75$\pm$0.8	49.76$\pm$0.9	49.75$\pm$0.9
	AT BNN	67.02$\pm$0.6	51.87$\pm$0.9	36.76$\pm$0.9	25.06$\pm$0.7	51.81$\pm$0.9	51.80$\pm$0.9	51.81$\pm$0.9	51.80$\pm$0.9
	Pretrained AT	70.55$\pm$0.7	52.25$\pm$0.7	37.11$\pm$0.8	24.03$\pm$0.7	52.19$\pm$0.7	52.18$\pm$0.7	52.16$\pm$0.7	52.13$\pm$0.7
	ADMM	65.94$\pm$0.4	47.37$\pm$0.6	32.77$\pm$0.6	22.45$\pm$0.5	47.34$\pm$0.5	47.31$\pm$0.5	47.30$\pm$0.4	47.31$\pm$0.5
	TRADES	69.54$\pm$0.2	52.08$\pm$0.7	34.54$\pm$0.3	23.01$\pm$0.4	52.03$\pm$0.5	52.01$\pm$0.6	52.03$\pm$0.5	52.02$\pm$0.5
	ANP-VS (ours)	**72.33$\pm$0.4**	**56.21$\pm$0.1**	**40.89$\pm$0.3**	**26.90$\pm$0.2**	**56.17$\pm$0.1**	**56.16$\pm$0.1**	**56.15$\pm$0.1**	**56.16$\pm$0.1**
CIFAR-100	Standard	11.68$\pm$0.7	2.81$\pm$0.2	0.91$\pm$0.1	0.41$\pm$0.0	2.77$\pm$0.1	2.76$\pm$0.2	2.76$\pm$0.1	2.75$\pm$0.1
	BP	13.01$\pm$0.3	3.12$\pm$0.1	1.04$\pm$0.1	0.47$\pm$0.0	3.02$\pm$0.3	3.06$\pm$0.2	3.03$\pm$0.2	3.04$\pm$0.1
	AT	35.10$\pm$0.9	19.07$\pm$0.8	10.62$\pm$0.4	5.89$\pm$0.8	19.06$\pm$0.8	19.03$\pm$0.8	19.02$\pm$0.8	19.04$\pm$0.7
	AT BNN	30.13$\pm$0.9	19.40$\pm$0.6	10.79$\pm$0.5	5.99$\pm$0.2	19.34$\pm$0.9	19.37$\pm$0.9	19.36$\pm$0.9	19.35$\pm$0.8
	Pretrained AT	33.92$\pm$0.2	19.86$\pm$0.6	11.39$\pm$0.3	6.27$\pm$0.1	19.85$\pm$0.5	19.84$\pm$0.6	19.81$\pm$0.6	19.83$\pm$0.6
	ADMM	34.59$\pm$0.2	19.65$\pm$0.5	10.50$\pm$0.3	4.77$\pm$0.4	19.60$\pm$0.3	19.57$\pm$0.3	19.59$\pm$0.3	19.59$\pm$0.2
	TRADES	33.89$\pm$0.1	21.22$\pm$0.3	10.80$\pm$0.1	4.51$\pm$0.2	21.15$\pm$0.4	21.16$\pm$0.3	21.20$\pm$0.4	21.15$\pm$0.4
	ANP-VS (ours)	**35.70$\pm$0.8**	**22.35$\pm$0.6**	**12.95$\pm$0.6**	**7.28$\pm$0.3**	**22.32$\pm$0.7**	**22.26$\pm$0.7**	**22.25$\pm$0.6**	**22.26$\pm$0.7**

Evaluation on CIFAR-10 and CIFAR-100. In order to show that ANP-VS is capable of scaling to more complex datasets, we evaluate our method on CIFAR-10 and CIFAR-100 dataset with VGG-16 architecture. We use $\varepsilon = 0.03$, 10 PGD steps for training and 40 steps with random restart for evaluation. The results are summarized in Table 1. We make the following observations from the results: (1) ANP-VS achieves $\sim 7\%$ improvement in adversarial accuracy, $\sim 58\%$ reduction in vulnerability of the network against white-box and black-box attack over all the baselines for both the datasets. (2) Our proposed method is not only effective on adversarial robustness but also outperforms all the compared adversarial baselines on the standard generalization performance, and strongly support our hypothesis of obtaining robust features. (3) Compared to the standard adversarial training baselines, our method shows a reduction $\sim 85\%$ in the memory-footprint with $2\times$ fewer FLOPS, demonstrating the effectiveness of our method.

One might also be concerned regarding the number of PGD steps and different ε values as for certain defenses; the robustness decreased as the number of PGD steps were increased (Engstrom et al., 2018). Table 2 shows the results for different ℓ_∞ epsilon values and PGD steps up to 1000. Remarkably, in both scenarios, our proposed defense outperforms all the compared baselines, illustrating the practicality of our method. These results indicate that even if the attacker uses greater resources to attack, the effect on our proposed method is negligible. This allows us to conclude that ANP-VS can induce reliable and effective adversarial robustness with much less computation cost.

5.3. Ablation studies

Analysis of individual components. To further gain insights into the performance of ANP-VS, we perform ablation studies to dissect the effectiveness of various components (ANP and VS) for robustness in Table 3. First, we study the impact of VS loss on AT, whether VS produces better robustness, and curtails the vulnerability of the network. One can observe that AT-VS improves the adversarial accuracy ($\sim 3\%$) and leads to $\sim 50\%$ reduction in vulnerability over AT for MNIST and CIFAR-10 dataset. In addition, note that the clean accuracy for AT-VS is approximately the same as AT across all the datasets, which indicates that VS may have a limited effect on standard accuracy, but has a substantial impact on robustness.

Second, we investigate the impact of ANP on AT, whether pruning the vulnerable latent-features is more effective than merely suppressing the vulnerability of those features. Table 3 shows that ANP not only improves the robustness ($\sim 10\%$) but also the standard generalization ($\sim 3\%$) over AT. Furthermore, we can observe that ANP on AT significantly outperforms VS on AT with CIFAR-10 dataset by $\sim 4\%$ on the adversarial accuracy while significantly reducing the computational cost. This observation supports our conjecture that ANP incorporates the vulnerability of latent-features, and pruning the vulnerable latent-features improves robustness. We know from these results that using VS or ANP can already outperform AT, while incorporating them together in ANP-VS additionally further boosts the performance, as testified.

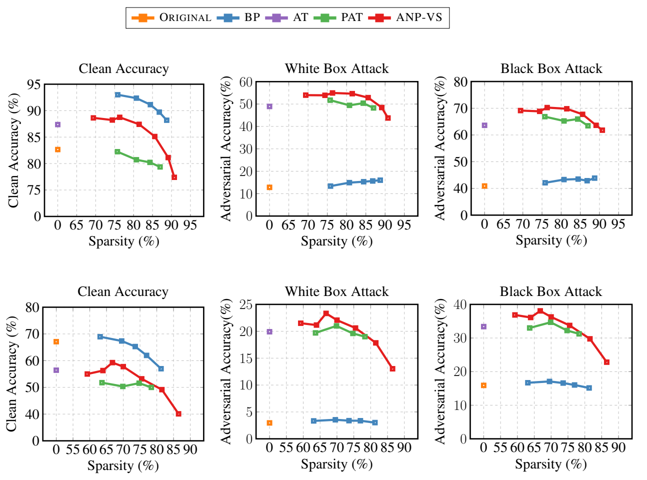


Figure 3. Comparison of clean and adversarial accuracy for different sparsity levels. Top: VGG-16 on CIFAR10. Bottom: VGG-16 on CIFAR100 dataset.

Table 3. Ablation studies for Vulnerability Suppression loss (VS) and Adversarial Neural Pruning (ANP) to investigate the impact of VS and ANP against ℓ_∞ white-box PGD attack. $\uparrow$ ($\downarrow$) indicates that the higher (lower) number is the better. The best results are highlighted in bold.

	Model	Clean Acc. ($\uparrow$)	Adv. acc ($\uparrow$)	Vulnerability ($\downarrow$)
MNIST	AT	99.14±0.02	88.03±0.7	0.045±0.001
	AT-VS	**99.24±0.05**	90.36±0.8	0.022±0.001
	ANP	98.64±0.07	90.12±0.5	0.020±0.002
	ANP-VS	99.05±0.08	**91.31±0.9**	**0.011**±0.002
CIFAR-10	AT	87.50±0.5	49.85±0.9	0.050±0.002
	AT-VS	87.44±0.5	51.52±0.4	0.024±0.002
	ANP	**88.36±0.4**	55.63±0.9	0.022±0.001
	ANP-VS	88.18±0.5	**56.21±0.1**	**0.016±0.000**
CIFAR-100	AT	57.79±0.8	19.07±0.8	0.079±0.003
	AT-VS	57.74±0.6	20.06±0.9	0.061±0.005
	ANP	58.47±1.02	22.20±1.1	0.037±0.003
	ANP-VS	**59.15±1.2**	**22.35±0.6**	**0.035±0.001**

Robustness and sparsity. Our proposed method could be vital when we want to obtain a lightweight yet robust network. To show that we can achieve both goals at once, we experiment with different scaling coefficient for the KL term in Equation (9) to obtain architectures with varying degrees of sparsity, whose details can be found in the supplementary material.

Figure 3 clearly shows that ANP-VS outperforms AT up to a sparsity level of $\sim 80\%$ for CIFAR-10 and CIFAR-100 after which there is a decrease in the robust and standard generalization. The results are not surprising, as it is an overall outcome of the model capacity reduction and the removal of the robust features. We further analyze a baseline PAT where we first perform Bayesian pruning, freeze the dropout mask, and then perform AT (see Figure 3). We can observe that PAT marginally improves the robustness over AT but loses on clean accuracy. Note that ANP-VS significantly outperforms PAT, supporting our hypothesis that just naive approach of AT over pruning can hurt performance.

5.4. Further analysis on defense performance

Vulnerability analysis. We conduct additional experiments to visualize the vulnerability of the latent-feature space to investigate the robust and vulnerable features. Figure 4 (Top) compares the vulnerability of various models. One can observe that the latent-features of the standard model are the most vulnerable, and the vulnerability decreases with the AT and is further suppressed by half with ANP-VS. Further, note that the latent features of our proposed method capture more local information of the objects and align much better with human perception.

In addition, Figure 4 (Bottom) shows the histogram of the vulnerability of latent-features for input-layer defined in Equation (1) for various methods. We consistently see that standard Bayesian pruning zeros out some of the distortions in the latent-features, and AT reduces the distortion level of all the latent-features. On the other hand, ANP-VS does both, with the largest number of latent-features with zero distortion and low distortion level in general. In this vein, we can say that ANP-VS demonstrates the effectiveness of our method as a defense mechanism by obtaining robust features utilizing pruning and latent vulnerability suppression.

Loss landscape visualization. It has been observed that adversarial training flatters the adversarial loss landscape and obfuscated gradients can often lead to bumpier loss landscapes (Engstrom et al., 2018). To investigate this problem of obfuscated gradients, we visualize the adversarial loss landscape of the baseline models and ANP-VS in Figure 5. We vary the input along a linear space defined by the sign of gradient where x and y-axes represent the perturbation added in each direction, and the z-axis represents the loss.

We can observe that the loss is highly curved in the vicinity of the data point x for the standard networks, which reflects that the gradient poorly models the global landscape. On the other hand, we observe that both sparsity and adversarial training make the loss surface smooth, with our model obtaining the smoothest surface. It demonstrates that our proposed method is not susceptible to the problem of gradients obfuscation. In addition, it indicates that suppressing the network vulnerability leads to robust optimization, uncovering some unexpected benefits of our proposed method over adversarial training.

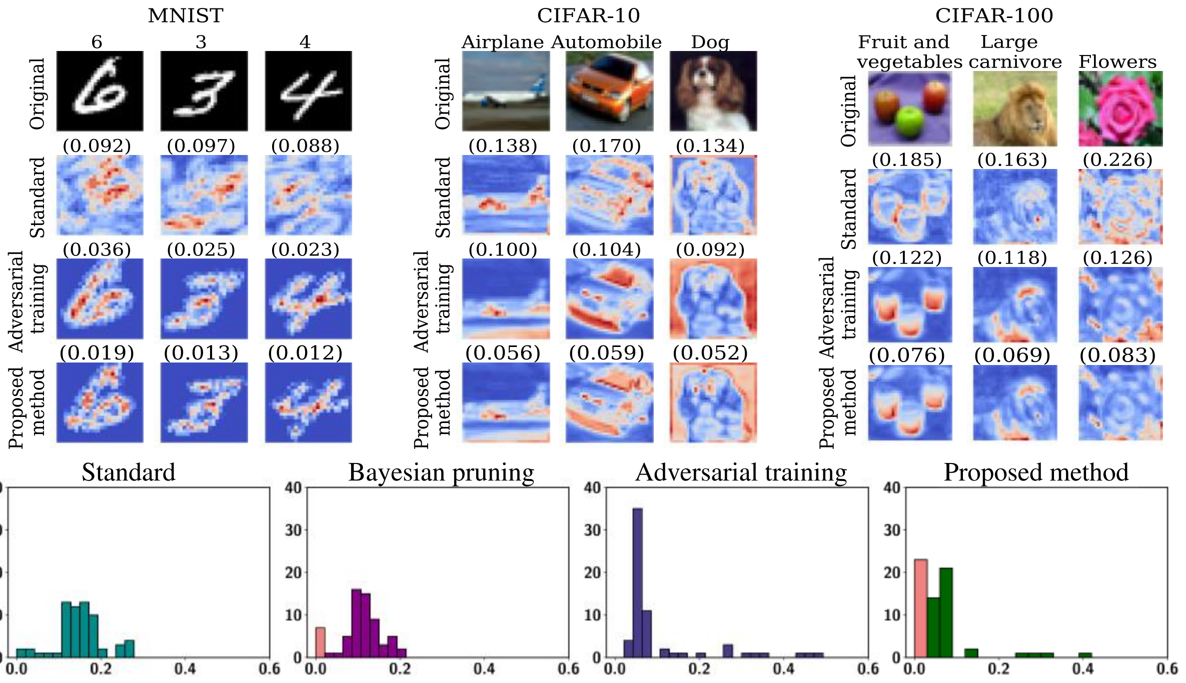


Figure 4. Top: Visualization of the vulnerability of the latent-features with respect to the input pixels for various set of datasets. Bottom: Histogram of vulnerability of the features for the input layer for CIFAR-10 with the number of zeros shown in orange color.

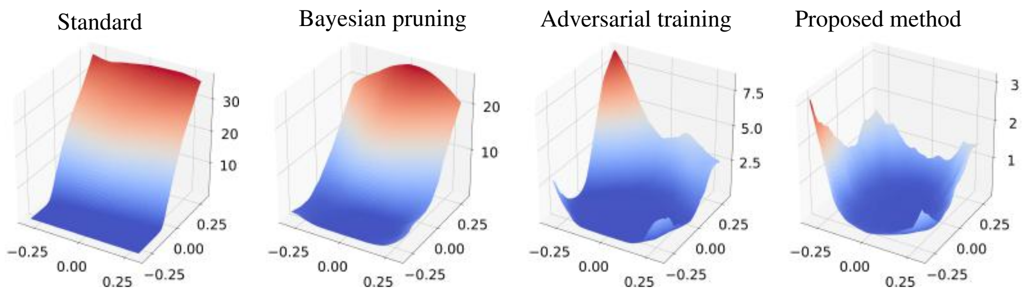


Figure 5. Comparison of loss landscapes for various methods. We observe that our proposed method results in a much smoother and flattened loss surface compared to adversarial training. The z axis represents the loss projected along two random directions.

6. Conclusion

We hypothesized that the adversarial vulnerability of deep neural networks comes from the distortion in the latent feature space since if they are suppressed at any layers of the deep network, they will not affect the prediction. Based on this hypothesis, we formally defined the vulnerability of a latent feature and proposed *Adversarial Neural Pruning*(ANP) as a defense mechanism to achieve adversarial robustness as well as a means of achieving a memory- and computation-efficient deep neural networks. Specifically, we proposed a Bayesian formulation that trains a Bayesian pruning (dropout) mask for adversarial robustness of the network. Then we introduced *Vulnerabiltiy Suppression (VS)* loss, that minimizes network vulnerability. To this end, we proposed *Adversarial Neural Pruning with Vulnerability Suppression (ANP-VS)*, which prunes the vulnerable features by learning pruning masks for them, to minimize the adversarial loss and feature-level vulnerability. We experimentally validate ANP-VS on three datasets against recent baselines, and the results show that it significantly improves the robustness of the deep network, achieving state-of-the-art results on all the datasets. Further qualitative analysis shows that our method obtains more interpretable latent features compared to standard counterparts, effectively suppresses feature-level distortions, and obtains smoother loss surface. We hope that our work leads to more follow-up works on adversarial learning that investigates the distortion in the latent feature space, which we believe is a more direct cause of adversarial vulnerability.

Acknowledgements

We thank the anonymous reviewers for their insightful comments and suggestions. We are also grateful to the authors of Lee et al. (2018) (Juho Lee, Saehoon Kim, and Hae Beom Lee)—for their insights and efforts replicating, extending and discussing our experimental results. This work was supported by Google AI Focused Research Award, Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2020-0-00153), Penetration Security Testing of ML Model Vulnerabilities and Defense), Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921), Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00075), and Artificial Intelligence Graduate School Program (KAIST). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

Athalye, A., Carlini, N., and Wagner, D. A. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018.

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Buckman, J., Roy, A., Raffel, C., and Goodfellow, I. Thermometer encoding: One hot way to resist adversarial examples. In *ICLR*, 2018.

Carlini, N. and Wagner, D. A. Towards evaluating the robustness of neural networks. *2017 IEEE Symposium on Security and Privacy (SP)*, 2016.

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. Parseval networks: Improving robustness to adversarial examples. In *ICML*, 2017.

Collins, M. D. and Kohli, P. Memory bounded deep convolutional networks. *arXiv preprint arXiv:1412.1442*, 2014.

Dai, B., Zhu, C., and Wipf, D. P. Compressing neural networks using the variational information bottleneck. In *ICML*, 2018.

Deng, J., Guo, J., Xue, N., and Zafeiriou, S. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*, 2019.

Dhillon, G. S., Azizzadenesheli, K., Bernstein, J. D., Kossaifi, J., Khanna, A., Lipton, Z. C., and Anandkumar, A. Stochastic activation pruning for robust adversarial defense. In *ICLR*, 2018.

Engstrom, L., Ilyas, A., and Athalye, A. Evaluating and understanding the robustness of adversarial logit pairing. *arXiv preprint arXiv:1807.10272*, 2018.

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., and Thrun, S. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 2017.

Gao, J., Wang, B., and Qi, Y. Deepmask: Masking DNN models for robustness against adversarial samples. *arXiv preprint arXiv:1702.06763*, 2017.

Garg, S., Sharan, V., Zhang, B., and Valiant, G. A spectral view of adversarially robust features. In *Neurips*, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *ICLR*, 2014.

Guo, Y., Zhang, C., Zhang, C., and Chen, Y. Sparse dnns with improved adversarial robustness. In *Neurips*, 2018.

Han, S., Pool, J., Tran, J., and Dally, W. J. Learning both weights and connections for efficient neural networks. In *Neurips*, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. Mask r-cnn. In *ICCV*, 2017.

Hendrycks, D., Lee, K., and Mazeika, M. Using pre-training can improve model robustness and uncertainty. In *ICML*, 2019.

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. Adversarial examples are not bugs, they are features. In *Neurips*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. In *Neurips*, 2015.

Krizhevsky, A. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Neurips*, 2012.

Kumaraswamy, P. A generalized probability density function for double-bounded random processes. *Journal of Hydrology*, 1980.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial machine learning at scale. In *ICLR*, 2016.

LeCun, Y. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Lee, J., Kim, S., Yoon, J., Lee, H. B., Yang, E., and Hwang, S. J. Adaptive network sparsification with dependent variational beta-bernoulli dropout. *arXiv preprint arXiv:1805.10896*, 2018.

Liu, X., Li, Y., Wu, C., and Hsieh, C.-J. Adv-BNN: Improved adversarial defense through robust bayesian neural network. In *ICLR*, 2019.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Molchanov, D., Ashukha, A., and Vetrov, D. Variational dropout sparsifies deep neural networks. In *ICML*, 2017.

Nalisnick, E. and Smyth, P. Stick-breaking variational autoencoders. *arXiv preprint arXiv:1605.06197*, 2016.

Neklyudov, K., Molchanov, D., Ashukha, A., and Vetrov, D. P. Structured bayesian pruning via log-normal multiplicative noise. In *Neurips*, 2017.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, 2016.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017.

Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Madry, A. Adversarially robust generalization requires more data. In *Neurips*, 2018.

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

Song, Y., Kim, T., Nowozin, S., Ermon, S., and Kushman, N. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *ICLR*, 2018.

Strm, N. Sparse connection and pruning in large dynamic artificial neural networks, 1997.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. In *ICLR*, 2013.

Tishby, N., Pereira, F. C., and Bialek, W. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. Robustness may be at odds with accuracy. In *ICLR*, 2019.

Uesato, J., O'Donoghue, B., Kohli, P., and van den Oord, A. Adversarial risk and the dangers of evaluating against weak attacks. In *ICML*, 2018.

Wang, L., Ding, G. W., Huang, R., Cao, Y., and Lui, Y. C. Adversarial robustness of pruned neural networks. *ICLR Workshop Submission*, 2018.

Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. Learning structured sparsity in deep neural networks. In *Neurips*, 2016.

Xu, W., Evans, D., and Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017a.

Xu, W., Evans, D., and Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017b.

Ye, S. and Xu, K. Adversarial robustness vs. model compression, or both. In *ICCV*, 2019.

Ye, S., Wang, S., Wang, X., Yuan, B., Wen, W., and Lin, X. Defending dnn adversarial attacks with pruning and logits augmentation. *ICLR Workshop Submission*, 2018.

Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., and Jordan, M. I. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019.

A. Adversarial variational information bottleneck

In this section, we extend the idea of Adversarial Neural Pruning to Variational Information Bottleneck (VIB). Variational information bottleneck (Dai et al., 2018) uses information theoretic bound to reduce the redundancy between adjacent layers. Let $p(\boldsymbol{h_i}|\boldsymbol{h_{i-1}})$ define the conditional probability and $I(\boldsymbol{h_i};\boldsymbol{h_{i-1}})$ define the mutual information between hidden layer activations $\boldsymbol{h_i}$ and $\boldsymbol{h_{i-1}}$ for every hidden layer in the network. For every hidden layer $\boldsymbol{h_i}$, we would like to minimize the information bottleneck (Tishby et al., 2000) $I(\boldsymbol{h_i};\boldsymbol{h_{i-1}})$ to remove interlayer redundancy, while simultaneously maximizing the mutual information $I(\boldsymbol{h_i};\boldsymbol{y})$ between $\boldsymbol{h_i}$ and the output $\boldsymbol{y}$ to encourage accurate predictions of adversarial examples. The layer-wise energy $\mathcal{L}_i$ can be written as:

$$\mathcal{L}_i = \beta_i I(\boldsymbol{h_i};\boldsymbol{h_{i-1}}) - I(\boldsymbol{h_i};\boldsymbol{y}) \tag{10}$$

The output layer approximates the true distribution $p(\boldsymbol{y}|\boldsymbol{h_L})$ via some tractable alternative $q(\boldsymbol{y}|\boldsymbol{h_L})$. Using variational bounds, we can invoke the upper bound as:

$$\begin{aligned}\mathcal{L}_i = \beta_i \mathbb{E}_{\boldsymbol{h_{i-1}}\sim p(\boldsymbol{h_{i-1}})}[\mathrm{D_{KL}}[p(\boldsymbol{h_i}|\boldsymbol{h_{i-1}})||q(\boldsymbol{h_i})]] - \\ \mathbb{E}_{\{\boldsymbol{x},\boldsymbol{y}\}\sim D, h\sim p(\boldsymbol{h}|\tilde{\boldsymbol{x}})}[\log q(\boldsymbol{y}|\boldsymbol{h_L})] \geq \mathcal{L}_i\end{aligned} \tag{11}$$

$\mathcal{L}_i$ in Equation 11 is composed of two terms, the first is the KL divergence between $p(\boldsymbol{h_i}|\boldsymbol{h_{i-1}})$ and $q(\boldsymbol{h_i})$, which approximates information extracted by $\boldsymbol{h_i}$ from $\boldsymbol{h_{i-1}}$ and the second term represents constancy with respect to the adversarial data distribution. In order to optimize Equation 11, we can define the parametric form for the distributions $p(\boldsymbol{h_i}|\boldsymbol{h_{i-1}})$ and $q(\boldsymbol{h_i})$ as follow:

$$\begin{aligned}p(\boldsymbol{h_i}|\boldsymbol{h_{i-1}}) &= \mathcal{N}(\boldsymbol{h_i}; f_i(\boldsymbol{h_{i-1}}) \odot \mu_i, \mathrm{diag}[f_i(\boldsymbol{h_{i-1}})^2 \odot \sigma_i^2] \\ q(\boldsymbol{h_i}) &= \mathcal{N}(\boldsymbol{h_i}; 0, \mathrm{diag}[\boldsymbol{\xi_i}])\end{aligned} \tag{12}$$

where $\boldsymbol{\xi}_i$ is an unknown vector of variances that can be learned from data. The gaussian assumptions help us to get an interpretable, closed-form approximation for the KL term from Equation 11, which allows us to directly optimize $\boldsymbol{\xi}_i$ out of the model.

$$\begin{aligned}\mathbb{E}_{\boldsymbol{h_{i-1}}\sim p(\boldsymbol{h_{i-1}})}[\mathrm{D_{KL}}[p(\boldsymbol{h_i}|\boldsymbol{h_{i-1}})||q(\boldsymbol{h_i})]] = \\ \sum_j \left[\log\left(1 + \frac{\mu_{i,j}^2}{\sigma_{i,j}^2}\right)\right]\end{aligned} \tag{13}$$

The final variational information bottleneck can thus be obtained using Equation 13:

$$\begin{aligned}\mathcal{L} = \sum_{i=1}^{L} \beta_i \sum_{j=1}^{r_i} \left[\log\left(1 + \frac{\mu_{i,j}^2}{\sigma_{i,j}^2}\right)\right] - \\ \mathbb{E}_{\{\boldsymbol{x},\boldsymbol{y}\}\sim D, \boldsymbol{h}\sim p(\boldsymbol{h}|\tilde{\boldsymbol{x}})}[\log q(\boldsymbol{y}|\boldsymbol{h_L})]\end{aligned} \tag{14}$$

where $\beta \geq 0$ is a coefficient that determines the strength of the bottleneck that can be defined as the degree to which we value compression over robustness.

B. Experiment setup

In this section, we describe our experimental settings for all the experiments. We follow the two-step pruning procedure where we pretrain all the networks using the standard-training procedure followed by network sparsification using various sparsification methods. We train each model with 200 epochs with a fixed batch size of 64. All the results are measured by computing mean and standard deviation across 5 trials upon randomly chosen seeds.

Our pretrained standard Lenet 5-Caffe baseline model reaches over 99.29% accuracy on MNIST and VGG-16 architecture reaches 92.76% and 67.44% on CIFAR-10 and CIFAR-100 dataset respectively after 200 epochs. We use Adam (Kingma & Ba, 2014) with the learning rate for the weights to be 0.1 times smaller than those for the variational parameters as in (Neklyudov et al., 2017; Lee et al., 2018). For Beta-Bernoulli Dropout, we set $\alpha/K = 10^{-4}$ for all the layers and prune the neurons/filters whose expected drop probability are smaller than a fixed threshold 10^{-3} as originally proposed in the paper. For Beta-Bernoulli Dropout, we scaled the KL-term by different values of trade-off parameter β where $\beta \in \{1, 4, 8, 10, 12\}$ for Lenet-5-Caffe and $\beta \in \{1, 2, 4, 6, 8\}$ for VGG-16. For Variational Information Bottleneck (VIBNet), we tested with trade-off parameter β in Equation 14 where $\beta \in \{10, 30, 50, 80, 100\}$ for Lenet-5-Caffe with MNIST and $\beta \in \{10^{-4}, 1, 20, 40, 60\}$ for VGG-16 with CIFAR-10 and CIFAR-100 dataset. For generating black-box adversarial examples, we used an adversarial trained full network for adversarial neural pruning and the standard base network for the standard Bayesian compression method.

C. More experimental results

Due to the length limit of our paper, some results are illustrated here.

C.1. Robustness of adversarial variational information bottleneck

The results for ANP-VS with Variational Information Bottleneck are summarized in Table 4. We can observe that ANP-VS with Variational Information Bottleneck significantly outperforms the base adversarial training for robustness of adversarial examples by achieving an improvement of $\sim 2\%$ in adversarial accuracy. Note that, ANP-VS leads to $\sim 50\%$ and $\sim 25\%$ reduction in vulnerability for CIFAR-10 and CIFAR-100 dataset with memory and computation efficiency. We emphasize that ANP can similarly be ex-

Table 4. Robustness and compression performance for MNIST on Lenet-5-Caffe, CIFAR-10 and CIFAR-100 on VGG-16 architecture under ℓ_∞-PGD attack for ANP-VS with Variational Information Bottleneck. All the values are measured by computing mean and standard deviation across 5 trials upon randomly chosen seeds. The best results over adversarial baselines are highlighted in bold.

			Adversarial accuracy (↑)		Vulnerability (↓)		Computational efficiency		
	Model	Clean accuracy (↑)	White box attack	Black box attack	White box attack	Black box attack	Memory (↓)	xFLOPS (↑)	Sparsity (↑)
MNIST	Standard	99.29±0.02	0.00±0.0	8.02±0.9	0.129±0.001	0.113±0.000	100.0±0.00	1.00±0.00	0.00±0.00
	BP	99.32±0.04	5.66±0.4	15.47±0.3	0.091±0.001	0.078±0.001	4.34±0.34	9.39±0.25	82.46±0.61
	AT	99.14±0.02	88.03±0.7	94.18±0.8	0.045±0.001	0.040±0.000	100.0±0.00	1.00±0.00	0.00±0.00
	AT BNN	99.16±0.05	88.44±0.4	94.87±0.2	0.364±0.023	0.199±0.031	200.0±0.00	0.50±0.00	0.00±0.00
	Pretrained AT	**99.18±0.06**	88.26±0.6	94.49±0.7	0.412±0.035	0.381±0.029	100.0±0.00	1.00±0.00	0.00±0.00
	ADMM	99.01±0.02	88.47±0.4	94.61±0.7	0.041±0.002	0.038±0.001	100.00±0.00	1.00±0.00	80.00±0.00
	TRADES	99.07±0.04	89.67±0.4	95.04±0.6	0.037±0.001	0.033±0.001	100.0±0.00	1.00±0.00	0.00±0.00
	ANP-VS (ours)	98.86±0.02	**90.11±0.9**	**95.14±0.8**	**0.017±0.001**	**0.015±0.001**	**4.87±0.21**	**10.06±0.87**	**78.48±0.42**
CIFAR-10	Standard	92.76±0.1	13.79±0.8	41.65±0.9	0.077±0.001	0.065±0.001	100.0±0.00	1.00±0.00	0.00±0.00
	BP	92.73±0.1	12.28±0.3	76.35±0.8	0.035±0.002	0.032±0.001	12.38±0.12	2.38±0.0.05	76.35±0.23
	AT	87.50±0.5	49.85±0.9	63.70±0.6	0.050±0.002	0.047±0.001	100.0±0.00	1.00±0.00	0.00±0.00
	AT BNN	86.69±0.5	51.87±0.9	64.92±0.9	0.267±0.013	0.238±0.011	200.0±0.00	0.50±0.00	0.00±0.00
	Pretrained AT	87.50±0.4	52.25±0.7	66.10±0.8	0.041±0.002	0.036±0.001	100.0±0.00	1.00±0.00	0.00±0.00
	ADMM	78.15±0.7	47.37±0.6	62.15±0.8	0.034±0.002	0.030±0.002	100.00±0.00	1.00±0.00	75.00±0.00
	TRADES	80.33±0.5	52.08±0.7	64.80±0.5	0.045±0.001	0.042±0.005	100.0±0.00	1.00±0.00	0.00±0.00
	ANP-VS (ours)	**87.56±0.2**	**53.41±0.5**	**68.12±0.7**	**0.025±0.002**	**0.021±0.001**	**12.09±0.26**	**2.43±0.02**	**77.02±0.32**
CIFAR-100	Standard	67.44±0.7	2.81±0.2	14.94±0.8	0.143±0.007	0.119±0.005	100.0±0.00	1.00±0.00	0.00±0.00
	BP	69.09±0.5	2.73±0.3	19.53±0.4	0.084±0.001	0.073±0.001	18.46±0.42	1.95±0.03	63.84±0.62
	AT	57.79±0.8	19.07±0.8	32.47±1.4	0.079±0.003	0.071±0.003	100.0±0.00	1.00±0.00	0.00±0.00
	AT BNN	53.75±0.7	19.40±0.6	30.38±0.2	0.446±0.029	0.385±0.051	200.0±0.00	0.50±0.00	0.00±0.00
	Pretrained AT	57.14±0.9	19.86±0.6	35.42±0.4	0.071±0.001	0.065±0.002	100.0±0.00	1.00±0.00	0.00±0.00
	ADMM	52.52±0.5	19.65±0.5	31.30±0.3	0.060±0.001	0.056±0.001	100.00±0.00	1.00±0.00	65.00±0.00
	TRADES	56.70±0.7	21.21±0.3	32.81±0.6	0.065±0.003	0.060±0.003	100.0±0.00	1.00±0.00	0.00±0.00
	ANP-VS (ours)	**59.77±0.4**	**21.53±0.6**	**36.82±0.7**	**0.048±0.002**	**0.042±0.002**	**16.46±0.34**	**2.06±0.02**	**67.19±0.57**

tended to any existing or future sparsification method to improve performance. Table 5 further shows the number of units for the baselines and our proposed method.

C.2. Features vulnerability

Figure 6 shows the histogram of the feature vulnerability for various datasets. We consistently observe that standard Bayesian pruning zeros out some of the distortions, AT reduces the distortion level of all the features and ANP-VS does both, with the most significant number of features with zero distortion and low distortion level in general which confirms that our proposed method works successfully as a defense against adversarial attacks. All these results overall confirm the effectiveness of our defense.

C.3. Features visualization

One might also be curious about the representation of the robust and vulnerable features in the latent-feature space. We visualize the robust and vulnerable features based on the vulnerability of a feature in the latent-feature space from our paper. Figure 7 shows the visualization of robust and vulnerable features in the latent space for adversarial training. Note that, AT contains features with high vulnerability (vulnerable feature) and features with less vulnerability (robust feature), which aligns with our observation that the latent features have a varying degree of susceptibility to adversarial perturbations to the input. As future work, we plan to explore more effective ways to suppress perturbation at the intermediate latent features of deep networks.

	Model	No of neurons
MNIST	Standard	20 – 50 – 800 – 500
	BP (BBD)	14 – 21 – 150 – 49
	BP (VIB)	12 – 19 – 160 – 37
	AT	20 – 50 – 800 – 500
	ANP-VS (BBD)	7 – 21 – 147 – 46
	ANP-VS (VIB)	10 – 23 – 200 – 53
CIFAR-10	Standard	64 – 64 – 128 – 128 – 256 – 256 – 256 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512
	BP (BBD)	57 – 59 – 127 – 101 – 150 – 71 – 31 – 41 – 35 – 10 – 46 – 48 – 16 – 16 – 25
	BP (VIB)	49 – 56 – 106 – 92 – 157 – 74 – 26 – 43 – 32 – 10 – 39 – 40 – 7 – 7 – 13
	AT	64 – 64 – 128 – 128 – 256 – 256 – 256 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512
	ANP-VS (BBD)	42 – 57 – 113 – 96 – 147 – 68 – 25 – 37 – 27 – 9 – 39 – 40 – 13 – 13 – 12
	ANP-VS (VIB)	40 – 57 – 104 – 93 – 174 – 96 – 30 – 48 – 39 – 9 – 49 – 57 – 10 – 10 – 12
CIFAR-100	Standard	64 – 64 – 128 – 128 – 256 – 256 – 256 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512
	BP (BBD)	62 – 64 – 128 – 123 – 244 – 203 – 84 – 130 – 95 – 18 – 152 – 157 – 32 – 32 – 101
	BP (VIB)	52 – 64 – 119 – 116 – 229 – 179 – 83 – 99 – 71 – 17 – 107 – 110 – 12 – 11 – 49
	AT	64 – 64 – 128 – 128 – 256 – 256 – 256 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512
	ANP-VS (BBD)	60 – 64 – 126 – 122 – 235 – 185 – 77 – 128 – 101 – 17 – 165 – 177 – 35 – 35 – 45
	ANP-VS (VIB)	44 – 58 – 110 – 109 – 207 – 155 – 81 – 86 – 66 – 19 – 88 – 86 – 15 – 15 – 36

Table 5. Distribution of neurons for all the layers of Lenet-5 Caffe for MNIST and VGG-16 architecture for CIFAR-10 and CIFAR-100 datasets.

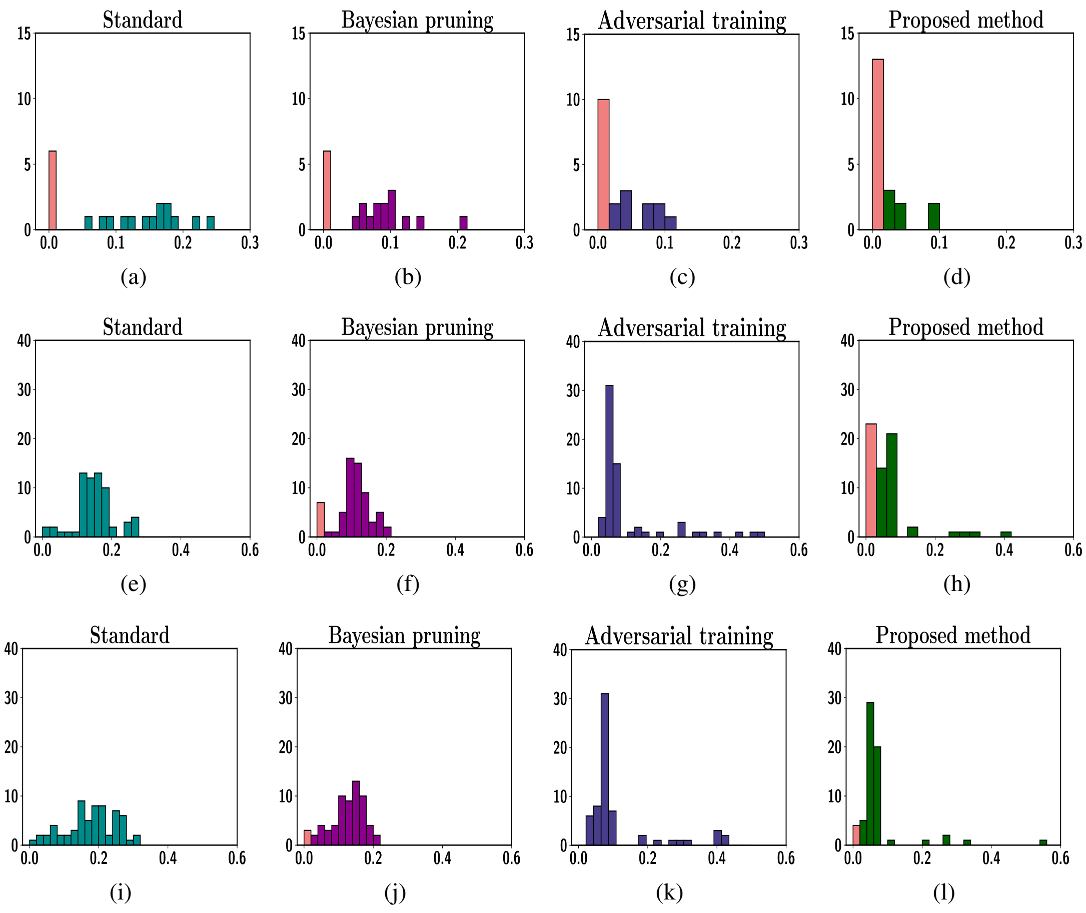
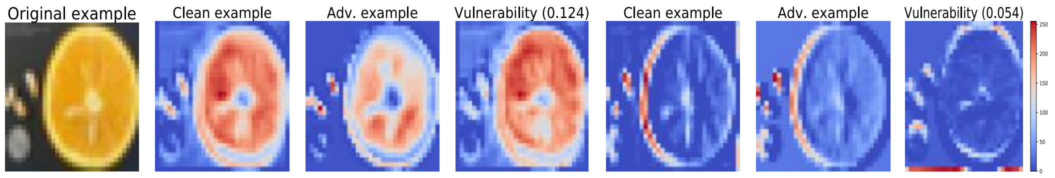


Figure 6. Histogram of vulnerability of the features for the input layer for MNIST in the top row, CIFAR-10 in the middle and CIFAR-100 in the bottom with the number of zeros shown in orange color.



(a) (b) (c) (d) (e) (f) (g)

Figure 7. Visualization of convolutional features of first layer of adversarial trained VGG-16 network with CIFAR-100 dataset. **b) - d)** represents the vulnerable latent-feature with high vulnerability (vulnerable feature) on b) clean example, c) Adversarial example d) Vulnerability (difference between clean and adversarial example) **e) - f)** represents the vulnerable latent-feature with low vulnerability (robust feature) on e) clean example, f) Adversarial example g) Vulnerability (difference between clean and adversarial example)