Adversarial Neural Pruning

Divyam Madaan¹, Sung Ju Hwang¹,²
KAIST¹, AITRICS²
{dmadaan, sjhwang82}@kaist.ac.kr

Abstract

It is well known that neural networks are susceptible to adversarial perturbations and are also computationally and memory intensive which makes it difficult to deploy them in real-world applications where security and computation are constrained. In this work, we aim to obtain both robust and sparse networks that are applicable to such scenarios, based on the intuition that latent features have a varying degree of susceptibility to adversarial perturbations. Specifically, we define vulnerability at the latent feature space and then propose a Bayesian framework to prioritize features based on their contribution to both the original and adversarial loss, to prune vulnerable features and preserve the robust ones. Through quantitative evaluation and qualitative analysis of the perturbation to latent features, we show that our sparsification method is a defense mechanism against adversarial attacks and the robustness indeed comes from our model’s ability to prune vulnerable latent features that are more susceptible to adversarial perturbations.

1 Introduction

In the last many years, deep-learning has achieved impressive results on diverse tasks such as face recognition [1], object recognition [2], playing video games [3] and many more. With the continued engenderment of more powerful models, there is a considerable increase in the number of parameters and the corresponding number of multiply-accumulate operations which makes it difficult for these models to be deployed on embedded and mobile platforms where we have constrained memory and computation. To address this issue, there exist many "pruning" or "sparsifying strategies" [4,5,6,7,8] which compress the model and help reduce the computational expense of the model without accuracy loss.

On the other hand, deep-learning models are also susceptible to adversarial perturbations that have been intentionally optimized to cause misclassification. 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, leading to successive rounds of claimed-successful defenses [9,10,11,12,13], followed by successful attacks [14,15,16,17] designed in light of the new defense. Both these problems give rise to some natural questions: Can we obtain robust and sparse models?

To answer this question, we need to draw a link between sparsity and adversarial robustness. Previous works have tried to answer the question in various ways, leading to different conclusions. It has been shown that sparsity leads to more robustness [18,19] whereas it has also been shown that robustness decreases with sparsity [20]. In this work, we propose the answer to this question from a completely different and novel perspective. We study the problem in the intermediate latent feature space of deep neural network models, based on the observation that the latent features have varying degree of susceptibility to adversarial perturbations to the input. Thus, some latent features amplify the perturbations at the input level while others remain relatively static. We formally define the vulnerability of the latent features, and define robust and vulnerable features depending on whether the perturbation on the latent features is greater than some value \( \delta \).
This definition of robust and vulnerable features in the latent feature space allows us to use sparsification as a defense mechanism, since any sparsification will effectively suppress the perturbation at the pruned feature by setting their values to zero. This is shown in Figure 1(a), where sparse networks are shown to have much smaller degree of mean distortion (average perturbation in the latent features across all layers). However, naive sparsification approaches will prune both the robust and vulnerable features, which will limit its effectiveness as a defense mechanism, and when the sparsity is pushed further, it will prune out robust features which will hurt the model robustness. To tackle this limitation, we propose a novel adversarial neural pruning method that adversarially learns the pruning mask, such that we can prune out vulnerable features while preserving robust ones. Our method requires little or no modification of the existing network architectures, can be applied to any pretrained networks and it effectively suppresses the distortion in the latent feature space (See Figure 1) and thus obtains a model that is more robust to adversarial perturbations. We validate our model on multiple heterogeneous datasets, namely MNIST, CIFAR-10, and CIFAR-100 with different base networks for its adversarial robustness, whose results show that our method obtains significantly improved accuracy even with adversarial attacks, while at the same time obtains compact networks.

The contribution of this work is threefold:

- We consider the vulnerability of latent features as the main cause of neural networks’ susceptibility to adversarial attacks, and formally describe the concepts of vulnerable and robust latent features, based on the expectation of the distortion w.r.t. input perturbations.
- We explore the relationship between sparsity and robustness and show that while sparsity improves the robustness of the given neural networks by zeroing out distortion at the pruned features, it is still orthogonal to robustness and even degenerates robustness at high degree, via experimental results and visualization of the loss landscape.
- Motivated by the above findings, we propose a novel Adversarial Neural Pruning method that prunes out vulnerable features while preserving robust ones, by adversarially learning the pruning mask in a Bayesian framework, which achieves state-of-the-art robustness on CIFAR-10 and CIFAR-100 datasets with large reduction in memory and computation.

2 Related Work

Adversarial robustness Since the literature on adversarial robustness of neural networks is vast, we only discuss some of the most relevant studies. Large number of defenses [21, 22, 10, 23, 24, 25] have been proposed and consequently broken by more sophisticated attack methods [14, 15, 16, 17]. One of the most successful defense is adversarial training [9], in which the neural network is trained to optimize the maximum loss obtainable using projected gradient descent over the region of allowable perturbations, which improves upon the Iterative Method [26]. However, recently, Schott et al. [27] showed that even the adversarial training [9] trained with $l_{\infty}$ can be broken with $l_0$-attacks. There has also been previous work which considered robust and vulnerable features at the input level. Garg et al. [28] establish a relation between adversarially robust features and the spectral property of the geometry of the dataset and Gao et al. [29] proposed to remove unnecessary features in order to get robustness. Our work is different from these existing work in that we consider and define the vulnerability at the latent feature level, which is more directly related to 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 [30, 31]; however, such heuristics-based pruning often degenerates accuracy, and Han et al. [32] proposed an iterative retraining and pruning approach to recover from the damage from pruning. Using sparsity-inducing regularization (e.g. $\ell_1$) is another popular approach for network sparsification. However elementwise sparsity does not yield practical speed-ups and Wei et al. [33] proposed to use group sparsity to drop a neuron or a filter as a whole, that will reduce the actual network size. Malchanov et al. [34] proposed to learn the individual dropout rates per weight with sparsity-inducing priors to completely drop out unnecessary weights, and Neklydov et al. [4] proposed to exploit structured sparsity by learning masks for each neuron or filter. Lee et al. [8] 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. [7] which minimizes the information theoretic bound to reduce the redundancy between layers.

Robustness and sparsity The sparsity and robustness have been explored and modelled together in various recent works. Guo et al. [18] 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. [19] experimentally discuss how pruning shall effect robustness with a similar conclusion. In contrary, Wang et al. [20] derived opposite conclusions showing that robustness decreases with increase in sparsity. However, all these works test their hypothesis on heuristic pruning techniques without adversarial training. On the contrary, we sparsify networks while explicitly targeting for robustness, as we learn the pruning (dropout) mask to minimize loss on adversarial examples.

3 Robustness of Deep Representations

Before presenting adversarial neural pruning, we first briefly introduce the concept of robust and vulnerable features in the deep latent representation space. We denote dataset as $D = \{(x, y_i)\}$, $x_i \in \mathbb{R}$ with $(x_i, y_i)$ corresponding to a data-point and its label. Let $W$ be the weights parameterized by $\theta$ for a neural network $\Omega$ with $L$ layers, $\theta \in \Theta$ denote the vector of parameters of the neural network and $\{h_i\}_{i=1}^L$ where $h_i \in \mathbb{R}^{r_i}$ denote network hidden layer activations.

Vulnerability of a feature Vulnerability of a latent feature could be thought as the expectation of the absolute difference between the feature value for a clean example and its adversarial perturbation. This could be formally defined as follows:

$$v(z) = \mathbb{E}_{(x,y) \sim D}|z^{\text{clean}} - z^{\text{adv}}|,$$

where $z^{\text{clean}}$ is the feature value for a clean example $x^{\text{clean}}$ and $z^{\text{adv}}$ is the feature value for an adversarially perturbed example $x + \delta$ denoted by $x^{\text{adv}}$, and $v$ measures the vulnerability of that feature by measuring the distortion in a feature in the presence of an adversary.

Definition 1 A feature $z$: $X \to \mathbb{R}$ is said to be $(\epsilon, \delta)$-robust to adversarial perturbation for a distribution $D$ with respect to the vulnerability metric $v$, if for a given $\epsilon$ there exists $z^{\text{adv}}$ such that $v(z^{\text{clean}}, z^{\text{adv}}) < \delta$. Formally:

$$\mathbb{E}_{(x,y) \sim D}|z^{\text{clean}} - z^{\text{adv}}| < \delta$$

Definition 2 A feature $z$: $X \to \mathbb{R}$ is said to be $(\epsilon, \delta)$-vulnerable to adversarial perturbation for a distribution $D$ with respect to the vulnerability metric $v$, if for a given $\epsilon$ there exists $z^{\text{adv}}$ such that $v(z^{\text{clean}}, z^{\text{adv}}) \geq \delta$. Formally:

$$\mathbb{E}_{(x,y) \sim D}|z^{\text{clean}} - z^{\text{adv}}| \geq \delta$$

Figure 2 shows the visualization of robust and vulnerable features in the latent space for adversarial training. It is important to observe that adversarial training also contains features with high vulnerability (vulnerable feature) and features with less vulnerability (robust feature) which align with our observation that the latent features have a varying degree of susceptibility to adversarial perturbation to the input.
To measure the vulnerability of an entire network $\Omega$, we simply need to compute the sum of the vulnerability for $N$ features across all $L$ layers as follows:

$$V(\Omega) = \frac{1}{L} \sum_{j=1}^{L} \sum_{i=1}^{N} v_j(z_i)$$  

(2)

**Adversarial Training**  Adversarial Training [35, 26] was proposed as a data augmentation method to train the network on the mixture of clean and adversarial examples until the loss converges. Instead of using it as a data augmentation technique, Madry et al. [9] incorporated the adversarial search inside the training process by solving the following non-convex outer minimization problem and a non-concave inner maximization problem:

$$\arg\min_{\theta} \mathbb{E}_{(x, y) \sim D} \left[ \max_{\delta \in S} \mathcal{L}(\theta, x + \delta, y) \right]$$  

(3)

In case of adversarial training, it is able to distinguish between the robust and vulnerable features (See Figure 1). While the standard training results in obtaining vulnerable features, adversarial training enables to select the robust features which are essentially required by the model to attain adversarial robustness.

**Robustness and Sparsity**  To reduce the vulnerable features, we can further make use of various sparsification methods. Guo et al. [18] used weight pruning and activation pruning to show that sparsifying networks leads to more robust networks which is the effect of the above principle. The actual reason behind the robustness is obvious by our definitions: sparsity suppresses vulnerability to 0 and thus reduces the vulnerability of the network. Yet, the network still does not take into account the robustness of a feature. To address this limitation, we introduce adversarial neural pruning in the next section which is a novel method to prune these vulnerable features.

4 Adversarial Neural Pruning

In this section, we propose adversarial neural pruning which combines the idea of adversarial training [9] with the Bayesian pruning methods. The basic idea of adversarial neural pruning is that if we train the dropout mask on adversarial examples then we can make the model sparse by retaining the robust features. Let $\mathcal{L}(\theta \odot M, x, y)$ be the loss function at data point $x$ with class $y$ for the model with parameters $\theta$ and mask parameters $M$, we can use Projected Gradient Descent (PGD) [9] to generate the adversarial examples:

$$x^{t+1} = \Pi_{x+\Delta}(x^t + \alpha \text{sgn}(\nabla_x \mathcal{L}(\theta \odot M, x, y)))$$  

(4)

where $\alpha$ is the step size and $\text{sgn}(\cdot)$ returns the sign of the vector. In this work, we consider the $l_\infty$-bounded perturbations, i.e., $\Delta = \{x + \delta = ||\delta||_\infty \leq \epsilon \}$ where $\delta$ is the added perturbation from the set of allowed perturbations for each example.
We then use the following objectives to train the weight and mask parameters for our model:

\[
\mathcal{L}_\theta = \arg\min_\theta \left[ \mathbb{E}_{(x, y) \sim \mathcal{D}} \left( \mathcal{L}(\theta \odot M, x, y) \right) + \mathbb{E}_{(x, y) \sim \mathcal{D}} \left( \max_{\delta \in \mathcal{S}} \mathcal{L}(\theta \odot M, x + \delta, y) \right) \right]
\]

\[
\mathcal{L}_M = \arg\min_M \mathbb{E}_{(x, y) \sim \mathcal{D}} \left( \max_{\delta \in \mathcal{S}} \mathcal{L}(\theta \odot M, x + \delta, y) \right)
\]

We now introduce our proposed method Adversarial neural pruning for Variational information bottleneck [7] and Beta Bernoulli dropout [8] based on our complete algorithm [1]. We emphasize that our proposed method can be extended to any existing or new sparsification method in a similar way.

Adversarial Variational Information Bottleneck Variational information bottleneck [7] minimizes the information theoretic bound that reduces the redundancy between adjacent layers by aggregating useful information into a subset of neurons that can be preserved. Let \( p(h_i | h_{i-1}) \) define the conditional probability and \( I(h_i; h_{i-1}) \) define the mutual information between hidden layer \( h_i \) and \( h_{i-1} \) for the adversarial examples. The layer-wise energy \( \mathcal{L}_i \) and the final loss \( \mathcal{L} \) can then be defined as follows:

\[
\mathcal{L}_i = \gamma_i I(h_i; h_{i-1}) - I(h_i; y)
\]

\[
\mathcal{L} = \sum_{i=1}^{L} \gamma_i \sum_{j=1}^{r_i} \left[ \log \left( 1 + \frac{\mu_j^2}{\sigma_j} \right) \right] - \mathbb{E}_{(x, y) \sim \mathcal{D}, h \sim p(h|x_{adv})} [\log q(y|h_L)]
\]

where \( \gamma \) controls the degree of value compression over robustness, \( \mu \) and \( \sigma \) refer to the learnable parameters for the parametric form of the distribution \( q(y|h_L) \). Adversarial VIB minimizes the information bottleneck [36] \( I(h_i; h_{i-1}) \) for every hidden layer \( h_i \) which removes the vulnerable subset of neurons, while simultaneously maximizes the mutual information \( I(h_i; y) \) between \( h_i \) and the adversarial examples to encourage accurate predictions of the adversarial examples.

Adversarial Beta Bernoulli Dropout Beta Bernoulli Dropout [8] learns to set the dropout rate for each individual neuron, by generating the dropout mask from sparsity-inducing beta-Bernoulli prior to each individual neuron. Let \( z_n \in \{0, 1\}^K \) be the mask sampled from the finite-dimensional beta-Bernoulli prior to be applied for the \( n \)-th observation \( x_n \). Using the Stochastic Gradient Variational Bayes (SGVB) framework [37], we can get the final loss \( \mathcal{L} \) as:

\[
\mathcal{L} = \sum_{n=1}^{N} \mathbb{E}_q [\log p(y_n | f(x_{n, adv}; W))] - D_{KL}[q(Z; \pi) || p(Z; \pi)]
\]

where

\[
D_{KL}[q(Z; \pi) || p(Z; \pi)] = \sum_{k=1}^{K} \left\{ \frac{a_k - \alpha/K}{a_k} \left( -\gamma - \Psi(b_k) - \frac{1}{b_k} \right) + \log \frac{a_k b_k}{\alpha/K} - \frac{b_k - 1}{b_k} \right\}
\]

where \( \gamma \) is Euler-Mascheroni constant, \( \Psi(.) \) is digamma function and \( \pi \) is the Kumaraswamy distribution [38] with parameters \( a \) and \( b \). The first term in the loss measures the log-likelihood of the adversarial samples w.r.t. \( q(Z; \pi) \) and the second term regularizes \( q(Z; \pi) \) so it doesn’t deviate too much from the prior distribution. The detailed derivation for equation [6] and equation [7] is referred to the supplementary material.
Table 1: Comparison of different methods for MNIST on LeNet-5-Caffe, CIFAR-10 and CIFAR-100 on VGG-16 architectures. We report speedup in terms of Floating Point Operations (xFLOPs), memory and sparsity, robustness in terms of accuracy and vulnerability on adversarial examples generated from white-box and black-box attack.

| Data         | Model | Clean acc.(%) | xFLOPs | Memory(%) | Sparsity(%) | Adv. accuracy(%) | Vulnerability |
|--------------|-------|---------------|--------|-----------|-------------|------------------|--------------|
|              |       |               |        |           |             | White box Box    |              |
| MNIST        | Original | 99.29 | 1.0   | 100.0   | 0.00      | 8.02            | 1.705  | 1.304          |
|              | BBD    | 99.34 | 4.14  | 9.68    | 83.48     | 0.00            | 12.99 | 1.188          |
|              | VIB    | 99.32 | 4.34  | 9.39    | 82.46     | 5.66            | 15.47 | 2.976          |
|              | Adv. Train | 99.14 | 1.0   | 100.0   | 0.00      | 88.03           | 94.83 | 0.457          |
|              | Adv. BNN | 99.16 | 0.5   | 200.0   | 0.00      | 84.44           | 95.87 | 0.364          |
|              | Pretrained– | 99.18 | 1.0   | 100.0   | 0.00      | 88.26           | 94.49 | 0.412          |
|              | Adv. Train | 97.67 | 9.77  | 6.69    | 84.45     | 89.12           | 94.18 | 0.315          |
|              | Adv. VIB | 98.86 | 4.87  | 10.06   | 78.48     | 89.01           | 94.94 | 0.307          |
| CIFAR-10     | Original | 92.65 | 1.0   | 100.0   | 0.00      | 12.80           | 40.89 | 0.293          |
|              | BBD    | 93.00 | 2.36  | 12.39   | 75.82     | 13.36           | 39.14 | 0.220          |
|              | VIB    | 92.73 | 2.38  | 12.28   | 76.35     | 12.35           | 41.32 | 0.224          |
|              | Adv. Train | 87.37 | 1.0   | 100.0   | 0.00      | 48.93           | 63.61 | 0.125          |
|              | Adv. BNN | 86.54 | 0.5   | 200.0   | 0.00      | 51.59           | 64.20 | 0.315          |
|              | Pretrained– | 87.50 | 1.0   | 100.0   | 0.00      | 52.25           | 66.10 | 0.138          |
|              | Adv. Train | 88.74 | 2.33  | 12.46   | 76.39     | 55.31           | 70.30 | 0.090          |
|              | Adv. VIB | 87.41 | 2.43  | 12.09   | 76.50     | 52.13           | 67.95 | 0.077          |
| CIFAR-100    | Original | 67.10 | 1.0   | 100.0   | 0.00      | 2.95            | 15.90 | 0.343          |
|              | BBD    | 69.08 | 1.93  | 18.87   | 63.05     | 3.54            | 17.07 | 0.226          |
|              | VIB    | 69.09 | 1.95  | 18.46   | 63.84     | 2.73            | 19.53 | 0.312          |
|              | Adv. Train | 53.21 | 0.5   | 200.0   | 0.00      | 19.40           | 30.38 | 0.477          |
|              | Adv. BNN | 57.14 | 1.0   | 100.0   | 0.00      | 19.86           | 35.42 | 0.175          |
|              | Pretrained– | 59.52 | 2.00  | 16.87   | 66.63     | 23.23           | 37.99 | 0.130          |
|              | Adv. VIB | 59.77 | 2.02  | 17.55   | 64.75     | 20.43           | 36.31 | 0.137          |

5 Experiments

5.1 Datasets

1) MNIST. This dataset [39] contains 60,000 28 × 28 grey scale images of handwritten digits for training example images, 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 [40] 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 [41] for this dataset with 13 convolutional and two fully connected layers with pre-activation batch normalization and Binary Dropout.

3) CIFAR-100. This dataset [40] 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 [41] similar to CIFAR-10 as the base network for this dataset.

5.2 Baselines and our model

1) Original. The base convolution neural network.

2) BBD. The Base network with Beta-Bernoulli Dropout (BBD) [8].

3) VIB. The Base network with VIBNet sparsity criterion [7].

4) Adv. Train. The base adversarial trained convolution neural network [9].

5) Adv. BNN. The base adversarial trained bayesian neural network [13].

6) Pretrained Adv. Train. The base adversarial trained network on a pretrained base network.

7) Adv. BBD. The base network with adversarial neural pruning on BBDDropout.

8) Adv. VIB. The base network with adversarial neural pruning on VIBNet.

1https://github.com/BVLC/caffe/tree/master/examples/mnist
Figure 3: Comparison of clean and adversarial accuracy for adversarial examples generated from white-box and black-box attack for different sparsity levels for Top VGG-16 on CIFAR10 Bottom VGG-16 on CIFAR100. We can see that adversarial neural pruning results in state-of-the-art robustness and outperforms base methods even with high sparsity percentage. The adversarial and clean accuracy falls after a certain level due to the pruning of robust latent-features after a certain sparsity level.

We have compared our approach against arguably the largest number of competing methods designed to obtain robustness. All models and algorithms are implemented using the Tensorflow library. We show all the results averaged over five runs. The experimental settings and the standard deviation for the results can be found in the supplementary material.

5.3 Quantitative Evaluation

We validate our model with respect to three metrics for compression ratio and model complexity: i) Run-time memory footprint (Memory) - The ratio of space for storing hidden feature maps during run-time in pruned-network versus original model. ii) Floating point operations (xFLOPs) - The ratio of the number of floating point operations for original model versus pruned-network. iii) Model size (Sparsity) - The ratio of the number of zero units in original model versus compressed network. We measure the robustness of our model using our proposed vulnerability metric in equation 2 and accuracy on adversarial examples generated from white box and black box attack from an adv. trained full network for adversarial neural pruning and the base network for standard Bayesian compression methods.

Experiments on MNIST We compare our method with the various baselines using the LeNet 5-Caffe architecture on the MNIST dataset. We use the attack parameters as in Madry et al. [9]: total adversarial perturbation of 76.5/255 (0.3), perturbation per step of 2.55/255 (0.01), and 20 total attack steps with random restarts for training and 40 total steps with random restarts for evaluating the trained model. We trained a standard Lenet 5-Caffe baseline model which reaches over 99.29% accuracy after 200 epochs averaged across five runs. The results for different methods are shown in Table 1 where the standard BBDDropout and VIBNet marginally improve robustness as compared to the standard-base model but they are not able to defend against the adversarial perturbations. Adversarial Neural Pruning achieves the best performance across all the compression and robustness metrics with 93.31% decrease in memory with significant speedup. We also experimented with different trade-off parameter, whose results can be found in the supplementary material.

Experiments on CIFAR-10 and CIFAR-100 We further evaluate our method on CIFAR-10 and CIFAR-100 datasets with the VGG-16 architecture. We use total adversarial perturbation of (0.03),

2Code available at github.com/divyam3897/advPruning
perturbation per step of (0.007), and 10 total attack steps for training and 40 total steps with random restart for evaluating the trained models. As shown in Table 1, adversarial neural pruning achieves state of the art robustness for CIFAR-10 and CIFAR-100 for both white-box and black-box attack with 87.91% and 83.13% reduction in memory footprint along with 38.4% and 23.97% reduction in vulnerability as compared to adversarial training for CIFAR-10 and CIFAR-100 respectively. This result confirms the effectiveness of our adversarial neural pruning as a defense mechanism.

We also present results for different sparsity levels for CIFAR-10 and CIFAR-100 in Figure 3 using scaling coefficient \( \gamma \) in equation 6 and by scaling the KL term in ELBO in equation 7 for BBDropout to obtain architectures with different levels of sparsity. We observe that both Adversarial BBDropout and Adversarial VIB outperform adversarial training up to a sparsity level of 80% for both CIFAR-10 and CIFAR-100 after which there is a decrease in the robustness and the clean accuracy which is an overall outcome of the model capacity reduction and the removal of the robust features. The distributions of neurons for various datasets can be seen in the supplementary material.

5.4 Further analysis

In this subsection, we further perform quantitative and qualitative analysis of our model for its vulnerability in the latent feature space, compared to that of the existing methods, as well as visualize the landscape of the loss obtained using our method.

Vulnerability visualization We first visualize the vulnerability of the latent-feature space. Figure 4 shows the vulnerability for each image for various set of datasets and the vulnerability distribution for all the features for the input layer for CIFAR-10. The results clearly show that the latent-features of the standard model are the most vulnerable, and the vulnerability decreases with the adversarial training and further suppressed by half with adversarial neural pruning. Further, the latent features of our proposed method align much better with the human perception which also results in the interpretable gradients as observed in the previous work [44, 45]. The bottom row of Figure 4 shows the histogram of the feature vulnerability defined in equation 1 for various methods. We consistently see that standard Bayesian pruning zeros out some of the distortions, and adversarial training reduces the distortion level of all the features. On the other hand, adversarial neural pruning does both, with the largest number of features with zero distortion and low distortion level in general.
Figure 5: Comparison of loss landscapes for various methods. We see that adversarial neural pruning behaves similar to adversarial training and results in a much smooth and flattened loss surface compared to adversarial training. The z axis represents the loss projected along two random directions.

Loss landscape visualization We further visualize the loss surface of the baseline models and network obtained using our adversarial pruning technique 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. 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 most smooth surface.

6 Conclusion

We propose a novel adversarial neural pruning, as a defense mechanism to achieve adversarial robustness as well as a means of achieving a memory- and computation-efficient deep neural networks. We observe that the latent features of deep networks have a varying degree of vulnerability/robustness to the adversarial perturbations to the input and formally defined the vulnerability and robustness of a latent feature. This observation suggests that we can increase the robustness of the model by pruning out vulnerable latent features, and we show that sparsification thus leads to certain degree of robustness over the base network for this obvious reason. We further propose a Bayesian formulation that trains the pruning mask in an adversarial training, such that the obtained neurons are beneficial both for the accuracy of the clean and adversarial inputs. Experimental results on a range of architectures with multiple datasets demonstrate that our adversarial pruning is effective in improving the model robustness. Further qualitative analysis shows that adversarial pruning obtains more interpretable latent features compared to standard counterparts, suppresses feature-level distortions in general while zeroing out perturbations at many of them, and obtains smooth loss surface.

References

[1] Jiankang Deng, Jia Guo, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. CoRR, abs/1801.07698, 2018.
[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. CoRR, abs/1502.01852, 2015.
[3] OpenAI. Openai five. https://blog.openai.com/openai-five/, 2018.
[4] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. CoRR, abs/1510.00149, 2015.
[5] Suraj Srinivas and R. Venkatesh Babu. Generalized dropout. CoRR, abs/1611.06791, 2016.
[6] Kirill Neklyudov, Dmitry Molchanov, Arsenii Ashukha, and Dmitry P. Vetrov. Structured bayesian pruning via log-normal multiplicative noise. In NIPS, 2017.
[7] Bin Dai, Chen Zhu, and David P. Wipf. Compressing neural networks using the variational information bottleneck. CoRR, abs/1802.10399, 2018.
[8] Juho Lee, Saehoon Kim, Jaehong Yoon, Hae Beom Lee, Eunho Yang, and Sung Ju Hwang. ADAPTIVE NETWORK SPARSIFICATION VIA DEPENDENT VARIATIONAL BETA-BERNOULLI DROPOUT, 2019.
[9] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

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

[11] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *CoRR*, abs/1704.01155, 2017.

[12] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*, 2018.

[13] Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. Adv-BNN: Improved adversarial defense through robust bayesian neural network. In *International Conference on Learning Representations*, 2019.

[14] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.

[15] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *CoRR*, abs/1802.00420, 2018.

[16] Jonathan Uesato, Brendan O’Donoghue, Aäron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. *CoRR*, abs/1802.05666, 2018.

[17] Anish Athalye and Nicholas Carlini. On the robustness of the CVPR 2018 white-box adversarial example defenses. *CoRR*, abs/1804.03286, 2018.

[18] Yiwen Guo, Chao Zhang, Changshui Zhang, and Yurong Chen. Sparse dnns with improved adversarial robustness. *CoRR*, abs/1810.09619, 2018.

[19] Shaoxian Ye, Siyue Wang, Xiao Wang, Bo Yuan, Wujie Wen, and Xue Lin. Defending dnn adversarial attacks with pruning and logits augmentation, 2018.

[20] Luyu Wang, Gavin Weiguang Ding, Ruitong Huang, Yanshuai Cao, and Yik Chau Lui. Adversarial robustness of pruned neural networks, 2018.

[21] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *CoRR*, abs/1511.04508, 2015.

[22] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *CoRR*, abs/1704.01155, 2017.

[23] Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, Jean Kossaifi, Aran Khanna, Zachary C. Lipton, and Animashree Anandkumar. Stochastic activation pruning for robust adversarial defense. In *International Conference on Learning Representations*, 2018.

[24] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *International Conference on Learning Representations*, 2018.

[25] Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*, 2018.

[26] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *CoRR*, abs/1611.01236, 2016.

[27] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on MNIST. In *International Conference on Learning Representations*, 2019.

[28] Shivam Garg, Vatsal Sharan, Brian Zhang, and Gregory Valiant. A spectral view of adversarially robust features. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10138–10148. Curran Associates, Inc., 2018.

[29] Ji Gao, Beilun Wang, and Yanjun Qi. Deepmask: Masking DNN models for robustness against adversarial samples. *CoRR*, abs/1702.06763, 2017.
[30] Nikko Ström. Sparse connection and pruning in large dynamic artificial neural networks, 1997.

[31] Maxwell D. Collins and Pushmeet Kohli. Memory bounded deep convolutional networks. CoRR, abs/1412.1442, 2014.

[32] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In NIPS, 2015.

[33] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, Advances in Neural Information Processing Systems 29, pages 2074–2082. Curran Associates, Inc., 2016.

[34] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML’17, pages 2498–2507. JMLR.org, 2017.

[35] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. arXiv e-prints, page arXiv:1412.6572, Dec 2014.

[36] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. arXiv preprint physics/0004057, 2000.

[37] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28, pages 2575–2583. Curran Associates, Inc., 2015.

[38] P. Kumaraswamy. A generalized probability density function for double-bounded random processes. Journal of Hydrology, 46(1):79–88, 1980.

[39] Yann LeCun. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

[40] Alex Krizhevsky. Learning multiple layers of features from tiny images. University of Toronto, 05 2012.

[41] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2015.

[42] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI}’16), pages 265–283, 2016.

[43] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. CoRR, abs/1602.02697, 2016.

[44] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In International Conference on Learning Representations, 2019.

[45] Beomsu Kim, Junghoon Seo, and Taegyun Jeon. Bridging adversarial robustness and gradient interpretability. CoRR, abs/1903.11626, 2019.

[46] Eric Nalisnick and Padhraic Smyth. Stick-breaking variational autoencoders. arXiv preprint arXiv:1605.06197, 2016.

[47] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. CoRR, abs/1412.6980, 2015.
A Adversarial Variational Information Bottleneck

Variational information bottleneck [7] uses information theoretic bound to reduce the redundancy between adjacent layers. Consider a distribution $\mathcal{D}$ of $N$ i.i.d samples $(x, y)$ input to a neural network $\Omega$ with $L$ layers with the network hidden layer activations as $\{h_i\}_{i=1}^L$ where $h_i \in \mathbb{R}^{c_i}$. Let $p(h_l|h_{l-1})$ define the conditional probability and $I(h_l; h_{l-1})$ define the mutual information between $h_l$ and $h_{l-1}$ for every hidden layer in the network. For every hidden layer $h_l$, we would like to minimize the information bottleneck [36] $I(h_l; h_{l-1})$ to remove interlayer redundancy, while simultaneously maximizing the mutual information $I(h_l; y)$ between $h_l$ and the output $y$ to encourage accurate predictions of adversarial examples. The layer-wise energy $\mathcal{L}_i$ can be written as:

$$\mathcal{L}_i = \gamma_i I(h_i; h_{i-1}) - I(h_i; y)$$

where $\gamma \geq 0$ is a coefficient that determines the strength of the bottleneck that can be defined as the degree to which we value compression over robustness. The output layer approximates the true distribution $p(y|h_L)$ via some tractable alternative $q(y|h_L)$. Let $x^{adv}$ be the adversarial example for $l_\infty$-bounded perturbations. i.e. $\Delta = \{x + \delta = \|\delta\|_\infty \leq \epsilon\}$ where $\delta$ is the added perturbation from the set of allowed perturbations for each example. Using variational bounds, we can invoke the upper bound as:

$$\mathcal{L}_i = \gamma_i \mathbb{E}_{h_{i-1} \sim p(h_{i-1})}[KL[p(h_i|h_{i-1})||q(h_i)]] - \mathbb{E}_{x,y \sim D, h \sim p(h|x^{adv})}[\log q(y|h_L)] \geq \mathcal{L}_i$$

$\mathcal{L}_i$ in equation 9 is composed of two terms, the first is the KL divergence between $p(h_i|h_{i-1})$ and $q(h_i)$, which approximates information extracted by $h_i$ from $h_{i-1}$ and the second term represents constancy with respect to the adversarial data distribution. In order to optimize equation 9, we can define the parametric form for the distributions $p(h_i|h_{i-1})$ and $q(h_i)$ as follow:

$$p(h_i|h_{i-1}) = \mathcal{N}(h_i; f_i(h_{i-1}) \odot \mu_i, \text{diag}[f_i(h_{i-1})^2 \odot \sigma^2_i])$$
$$q(h_i) = \mathcal{N}(h_i; 0, \text{diag}[\xi_i])$$

where $\xi_i$ 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 9 which allows us to directly optimize $\xi_i$ out of the model.

$$\mathbb{E}_{h_{i-1} \sim p(h_{i-1})}[KL[p(h_i|h_{i-1})||q(h_i)]] = \sum_j \left[ \log \left( 1 + \frac{\mu^2_{i,j}}{\sigma^2_{i,j}} \right) \right]$$

The final variational information bottleneck can thus be obtained using equation 11

$$\mathcal{L} = \sum_{i=1}^L \gamma_i \sum_{j=1}^{r_i} \left[ \log \left( 1 + \frac{\mu^2_{i,j}}{\sigma^2_{i,j}} \right) \right] - \mathbb{E}_{x,y \sim D, h \sim p(h|x^{adv})}[\log q(y|h_L)]$$

B Adversarial Beta Bernoulli Dropout

Beta Bernoulli Dropout [8] learns to set the dropout rate for each individual neuron, by generating the dropout mask from sparsity-inducing beta-Bernoulli prior for each individual neuron. Let $W$ be the parameters for the neural network and let $z_n \in \{0, 1\}^K$ be the mask sampled from the finite-dimensional beta-Bernoulli prior to be applied for the $n$-th observation $x_n$. The generative process of the bayesian neural network can be modelled as:

$$W \sim N(0, \lambda I), \quad \pi \sim \prod_{k=1}^K \text{beta}(\pi_k; \alpha/K, 1)$$
$$z_n|\pi \sim \prod_{k=1}^K \text{Ber}(z_{n,k}; \pi_k), \quad W_n = z_n \odot W$$

For $\pi$ we use Kumaraswamy distribution [38] and $z_k$ is sampled by reparametrization with continuous relaxation following [8]:

$$q(\pi_k; a_k, b_k) = a_k b_k \pi_k^{a_k-1} (1 - \pi_k^{a_k})^{b_k-1}$$
$$z_k = \text{sgm} \left( \frac{1}{\tau} \left( \log \frac{\pi_k}{1 - \pi_k} + \log \frac{u}{1 - u} \right) \right)$$
where \( \tau \) is a temperature continuous relaxation, \( u \sim \text{unif}[0, 1] \), and \( sgm(x) = \frac{1}{1 + e^{-x}} \). The KL-divergence between the prior and the variational distribution in closed form can then be defined as follows [46, 8]:

\[
D_{KL}[q(Z, \pi) || p(Z, \pi)] = \sum_{k=1}^{K} \left\{ \frac{a_k - \alpha / K}{a_k} \left( -\gamma - \Psi(b_k) - \frac{1}{b_k} \right) + \log \frac{a_kb_k \alpha / K}{b_k - 1} \right\}
\]

(15)

where \( \gamma \) is Euler-Mascheroni constant and \( \Psi(.) \) is the digamma function. We can use Stochastic Gradient Variational Bayes (SGVB) [37] to minimize the KL divergence between the variational distribution \( q(Z, \pi) \) of known parametric form and the true posterior \( p(Z|\pi) \). As we know from the SGVB framework minimizing the KL is equal to maximizing the evidence lower bound which can be done as follows:

\[
L = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_q[\log p(y_n|f(x_n^{adv}; W))] - D_{KL}[q(Z; \pi) || p(Z|\pi)]
\]

(16)

where the first term measures the log-likelihood of the adversarial samples w.r.t. \( q(Z; \pi) \) and the second term regularizes \( q(Z; \pi) \) so it doesn’t deviate too much from the prior distribution.

### C 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 and show the results averaged over five runs.

Our pretrained standard Lenet-5-Caffe baseline model reaches over 99.29\% accuracy on MNIST and VGG-16 reaches 92.65\% and 67.1\% on CIFAR-10 and CIFAR-100 respectively after 200 epochs. We use Adam [47] with the learning rate for the weights to be 0.1 times smaller than those for the variational parameters as in [6, 8]. 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 in equation 16 by different values of trade-off parameter \( \gamma \) where \( \gamma \in \{1, 4, 8, 10, 12\} \) for Lenet-5-Caffe and \( \gamma \in \{1, 2, 4, 6, 8\} \) for VGG-16. For Variational Information Bottleneck (VIBNet), we tested with trade-off parameter \( \gamma \) in equation 12 where \( \gamma \in \{10, 30, 50, 80, 100\} \) for Lenet-5-Caffe with MNIST dataset and \( \gamma \in \{10^{-4}, 1, 20, 40, 60\} \) for VGG-16 with CIFAR-10 and CIFAR-100 dataset.

### D More Experimental results

Due to the length limit of our paper, some results are illustrated here. Table [Appendix-1] shows the results of our paper with mean and standard deviation averaged over five runs for the clean and adversarial accuracy for all the methods. Figure [D.2] shows the results for MNIST for different sparsity levels. The results are consistent with those obtained on the CIFAR-10 and CIFAR-100 where the adversarial neural pruning outperforms or gives comparable performance to adversarial trained base network even at high sparsity levels.

Table [Appendix-2] shows the number of units for all the baselines and our proposed method. Figure [D.1] shows the histogram of the feature vulnerability for various datasets. We can consistently observe that standard Bayesian pruning zeros out some of the distortions, adversarial training reduces the distortion level of all the features and adversarial neural pruning does both, with the largest number of features with zero distortion and low distortion level in general which confirms that our adversarial neural pruning works successfully as a defense against adversarial attacks.
Table Appendix-1: Comparison of different methods for MNIST on Lenet-5-Caffe, CIFAR-10 and CIFAR-100 on VGG-16 architectures. We report compression in terms of Floating Point Operations (FLOPs), memory and sparsity, robustness in terms of accuracy (average and stddev) and vulnerability on adversarial examples generated from white-box and black-box attack.

| Data    | Model  | Clean acc (%) | xFLOPs | Memory (%) | Sparsity (%) | Adversarial accuracy (%) | Vulnerability |
|---------|--------|---------------|--------|------------|--------------|--------------------------|---------------|
|         |        |               |        |            |              | White box                | Black box     |
| MNIST   | Original | 99.29 (0.02)  | 1.0    | 100.0      | 0.00         | 0.00 (0.00)              | 8.02 (1.66)   | 1.705 (1.304) |
|         | BBD     | 99.34 (0.05)  | 4.14   | 9.68       | 83.48        | 0.00 (0.00)              | 12.99 (0.47)  | 1.691 (1.188) |
|         | VIB     | 99.32 (0.06)  | 4.34   | 9.39       | 82.46        | 5.66 (1.12)              | 15.47 (0.89)  | 2.976 (1.265) |
|         | Adv. Train | 99.14 (0.02) | 1.0    | 100.0      | 0.00         | 88.03 (0.82)            | 94.83 (0.93)  | 0.457 (0.295) |
|         | Adv. BNN | 99.16 (0.06)  | 0.5    | 200.0      | 0.00         | 88.44 (0.46)            | 95.87 (0.21)  | 0.364 (0.199) |
|         | Pretrained-Adv. Train | 99.18 (0.05) | 1.0    | 100.0      | 0.00         | 88.26 (0.65)            | 94.49 (1.01)  | 0.412 (0.381) |
|         | Adv. BBD | 98.67 (0.13)  | 9.77   | 6.39       | 84.25        | 89.12 (1.25)            | 94.18 (0.98)  | 0.315 (0.208) |
|         | Adv. VIB | 98.86 (0.06)  | 4.87   | 10.06      | 78.48        | 89.01 (1.02)            | 94.94 (0.57)  | 0.307 (0.188) |
| CIFAR-10 | Original | 92.05 (0.18)  | 1.0    | 100.0      | 0.00         | 12.80 (0.81)            | 40.89 (1.12)  | 0.297 (0.202) |
|         | BBD     | 93.08 (0.19)  | 2.36   | 12.39      | 75.82        | 13.19 (0.83)            | 39.14 (0.47)  | 0.220 (0.207) |
|         | VIB     | 92.73 (0.14)  | 2.38   | 12.28      | 76.35        | 12.35 (0.79)            | 41.32 (0.89)  | 0.224 (0.225) |
|         | Adv. Train | 87.37 (0.69) | 1.0    | 100.0      | 0.00         | 48.93 (0.52)            | 63.61 (0.52)  | 0.125 (0.121) |
|         | Adv. BNN | 86.54 (0.74)  | 0.5    | 200.0      | 0.00         | 51.59 (0.81)            | 64.20 (0.50)  | 0.315 (0.284) |
|         | Pretrained-Adv. Train | 87.50 (0.43) | 1.0    | 100.0      | 0.00         | 52.25 (0.96)            | 66.10 (1.24)  | 0.138 (0.115) |
|         | Adv. VIB | 88.74 (0.12)  | 2.35   | 12.46      | 76.35        | 55.31 (1.13)            | 79.30 (0.98)  | 0.190 (0.085) |
| CIFAR-100 | Original | 67.10 (0.28)  | 1.0    | 100.0      | 0.00         | 2.95 (0.13)             | 11.76 (0.76)  | 0.343 (0.274) |
|         | BBD     | 69.08 (1.07)  | 1.93   | 18.87      | 63.05        | 3.54 (0.11)             | 17.07 (0.56)  | 0.226 (0.230) |
|         | VIB     | 69.09 (0.19)  | 1.95   | 18.46      | 63.84        | 2.73 (0.10)             | 19.53 (0.75)  | 0.312 (0.260) |
|         | Adv. Train | 56.43 (1.73) | 1.0    | 100.0      | 0.00         | 19.91 (0.43)            | 33.39 (0.15)  | 0.171 (0.142) |
|         | Adv. BNN | 53.21 (0.77)  | 0.5    | 200.0      | 0.00         | 19.40 (1.03)            | 30.38 (1.58)  | 0.477 (0.422) |
|         | Pretrained-Adv. Train | 57.14 (1.64) | 1.0    | 100.0      | 0.00         | 19.86 (0.43)            | 35.42 (0.66)  | 0.175 (0.145) |
|         | Adv. VIB | 59.52 (1.02)  | 2.00   | 16.87      | 66.83        | 25.23 (0.53)            | 37.99 (0.53)  | 0.130 (0.114) |
|         | Adv. BBD | 59.77 (1.81)  | 2.02   | 17.55      | 64.75        | 20.43 (0.55)            | 36.31 (0.66)  | 0.137 (0.121) |

Figure D.1: 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.
Table Appendix-2: Clean and adversarial accuracy for models trained with and without adversarial training with flops, memory and sparsity percentage reduction.

| Data     | Model | No of neurons |
|----------|-------|---------------|
|          | Baseline | 20 – 50 – 800 – 500 |
| MNIST    | BBD     | 14 – 21 – 150 – 49 |
|          | VIB     | 12 – 19 – 160 – 37 |
|          | Adv. Train | 20 – 50 – 800 – 500 |
|          | Adv. BBD | 7 – 21 – 147 – 46 |
|          | Adv. VIB | 10 – 23 – 200 – 53 |
| CIFAR10  | Baseline | 64 – 64 – 128 – 256 – 256 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 |
|          | BBD     | 57 – 59 – 127 – 101 – 150 – 71 – 31 – 41 – 35 – 10 – 46 – 48 – 16 – 16 – 25 |
|          | VIB     | 49 – 56 – 106 – 92 – 157 – 74 – 26 – 43 – 32 – 10 – 39 – 40 – 7 – 7 – 13 |
|          | Adv. Train | 64 – 64 – 128 – 256 – 256 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 |
|          | Adv. BBD | 42 – 57 – 113 – 96 – 147 – 68 – 25 – 37 – 27 – 9 – 39 – 40 – 13 – 13 – 12 |
|          | Adv. VIB | 40 – 57 – 104 – 93 – 174 – 96 – 30 – 48 – 39 – 9 – 49 – 57 – 10 – 10 – 12 |
| CIFAR100 | Baseline | 64 – 64 – 128 – 256 – 256 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 |
|          | BBD     | 62 – 64 – 128 – 123 – 244 – 203 – 84 – 130 – 95 – 18 – 152 – 157 – 32 – 32 – 101 |
|          | VIB     | 52 – 64 – 119 – 116 – 229 – 179 – 83 – 99 – 71 – 17 – 107 – 110 – 12 – 11 – 49 |
|          | Adv. Train | 64 – 64 – 128 – 256 – 256 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 – 512 |
|          | Adv. BBD | 60 – 64 – 126 – 122 – 235 – 185 – 77 – 128 – 101 – 17 – 165 – 177 – 35 – 35 – 45 |
|          | Adv. VIB | 44 – 58 – 110 – 109 – 207 – 155 – 81 – 86 – 66 – 19 – 88 – 86 – 15 – 15 – 36 |

Figure D.2: Clean accuracy and Robustness with varying sparsity for MNIST on Lenet-5-Caffe architecture.