Regularizing deep networks using efficient layerwise adversarial training

Swami Sankaranarayanan
UMIACS
University of Maryland
College Park, MD
swamiviv@umiacs.umd.edu

Arpit Jain
GE Global Research
Niskayuna, NY
Arpit.Jain@ge.com

Rama Chellappa
UMIACS
University of Maryland
College Park, MD
rama@umiacs.umd.edu

Ser Nam Lim
GE Global Research
Niskayuna, NY
limser@ge.com

Abstract

Adversarial training has been shown to regularize deep neural networks in addition to increasing their robustness to adversarial examples. However, its impact on very deep state of the art networks has not been fully investigated. In this paper, we present an efficient approach to perform adversarial training by perturbing intermediate layer activations and study the use of such perturbations as a regularizer during training. We use these perturbations to train very deep models such as ResNets and show improvement in performance both on adversarial and original test data. Our experiments highlight the benefits of perturbing intermediate layer activations compared to perturbing only the inputs. The results on CIFAR-10 and CIFAR-100 datasets show the merits of the proposed adversarial training approach. Additional results on WideResNets show that our approach provides significant improvement in classification accuracy for a given base model, outperforming dropout and other base models of larger size.

1 Introduction

Deep neural networks (DNNs) have shown tremendous success in several computer vision tasks in recent years [[7],[16],[10]]. However, seminal works on adversarial examples [[5], [21]] have shown that DNNs are susceptible to imperceptible perturbations at input and intermediate layer activations respectively. From an optimization perspective, they also showed that adversarial training can be used as a regularization approach while training the deep networks. However, the effect of such adversarial training techniques as regularizers for very deep networks has not been systematically explored, since the computational overhead of adversarial training does not justify the marginal improvements over conventional regularization techniques.

In this work, we propose a simple and efficient learning algorithm that uses adversarial perturbations of intermediate layer activations to provide a stronger regularization while improving the robustness of the deep network to adversarial data. We avoid the expensive step of explicitly generating adversarial examples at different layers but rather perturb the activations of the current batch by gradients accumulated from the activations of the previous batch, in a mini-batch setting. Even though these gradient directions are not adversarial at the input layer, we show that they are strongly adversarial when applied at the intermediate layers. By using gradients from inputs belonging to a different class than the current input, we ensure that adversarial perturbations in the intermediate layers are

directed towards an adversarial class and force the network to learn robust representations at each layer resulting in improved discriminability.

The proposed pipeline does not add any significant overhead during training and thus can be easily extended to performing adversarial training in very deep neural networks. Our approach complements dropout regularizer and achieves regularization beyond dropout. It avoids over-fitting and generalizes well by achieving significant improvement in performance on the test set. We show that the trained network is extremely robust against adversarial examples even when it is not explicitly trained with adversarial inputs. The intent of this paper is not to generate adversarial perturbations for standalone images but rather use these adversarial gradients to efficiently regularize training. We perform several ablative experiments to highlight the properties of the proposed approach for adversarial training and present results for deep networks such as VGG [18], ResNets [7] and state of the art models such as WideResNets [24] using standard datasets such as CIFAR-10 and CIFAR-100.

2 Related Work

Many approaches have been proposed to regularize the training procedure of very deep networks. Early stopping and statistical techniques like weight decay are commonly used to prevent overfitting. Specialized techniques such as DropConnect [22], Dropout [20] have been successfully applied with very deep networks such as ResNets. Faster convergence of such deep architectures was made possible by Batch Normalization (BN) [8]. One of the added benefit of BN was that the additional regularization provided during training even made dropout regularization unnecessary in some cases.

The work of Szegedy *et al.* [21] showed the existence of adversarial perturbations for computer vision tasks by solving a box-constrained optimization approach to generate these perturbations. They also showed that training the network by feeding back these adversarial examples regularizes the training and makes network resistant to adversarial examples. Due to a relatively expensive layerwise training procedure, their analysis was limited to small datasets and shallow networks. [5] proposed the fast gradient sign method to generate such adversarial examples. To perform adversarial training, they proposed a modified loss function to also account for loss from adversarial examples. They showed significant improvements in the network's response to adversarial examples and obtained a regularization performance beyond dropout. [12] proposes a virtual adversarial training framework and show its regularization benefits for relatively deep models, while taking three times the normal training time. [13] proposed an iterative approach to generate much stronger adversarial perturbations and also presented a score function to measure robustness of classifiers against these examples. Furthermore, recent approaches such as deep contrastive smoothing [6], distillation [14] and stability training [25] have focused solely on improving robustness of the deep models to adversarial inputs. In this work, we present an efficient layerwise approach to adversarial training and demonstrate its ability as a strong regularizer for very deep models beyond the specialized methods mentioned above, in addition to improving model robustness to adversarial inputs.

Recent theoretical works such as [[3], [4]] analyze the effect of random, semi-random and adversarial perturbations on classifier robustness. They presented fundamental upper bounds on the robustness of classifier which depends on factors such as curvature of decision boundary and distinguishability between class cluster centers. Wang *et al.* [23] introduce the notion of strong robustness for classifiers and point out that the differences between generalization and robustness by characterizing the topology of the learned classification function.

3 Analysis of Perturbations

In this section, we present our approach and highlight the differences between related methods that perform adversarial training. In addition, we perform a small scale experiment to study the properties of the proposed adversarial training by analyzing the singular value spectrum of the Jacobian. We visualize the impact of these perturbations on the intermediate layer activations and conclude by illustrating the connection to robust optimization-based approaches.

We start by defining some notation. Let $\{x_i\}_{i=1}^N$ denote the set of images and $\{y_i\}_{i=1}^N$ denote the set of labels. Let $f : x \in \mathbb{R}^m \mapsto y \in \mathbb{L}$ denote the classifier mapping that maps the image to a discrete label set, $\mathbb{L}$. In this work, f is modeled by a deep CNN unless specified otherwise. We denote the loss function of the deep network by $\mathcal{J}(\theta, x, y)$ where θ represents the network parameters and $\{x, y\}$ are the input and output respectively. The deep network consists of L layers and $\nabla_l \mathcal{J}(\theta^t, x^t, y^t)$ denotes the backpropagated gradient of the loss function at the output of the l^{th} layer at iteration t. In the above expression, $l = 0$ corresponds to the input layer and $l = L - 1$, the loss layer. Let x_l^t be the

input activation to the l^{th} layer and r_l^t represents the perturbation that is added to x_l^t. For clarity, we drop the subscript l when talking about the input layer.

3.1 Approaches to generate Adversarial perturbations

Previous works on adversarial training have observed that training the model with adversarial examples acts as a regularizer and improves the performance of the base network on the test data. [21] define adversarial perturbations r as a solution of a box-constrained optimization as follows: Given an input x and target label y, they intend to minimize $||r||_2$ subject to (1) $f(x+r) = y$ and (2) $x + r \in [0, 1]^m$. Note that, if $f(x) = y$, then the optimization is trivial (i.e. $r = 0$), hence $f(x) \neq y$. While the exact minimizer is not unique, they approximate it using a box-constrained L-BFGS. More concretely, the value of c is found using line-search for which the minimizer of the following problem satisfies $f(x + r) = \hat{y}$, where $\hat{y} \neq y$:

$$\underset{r}{\text{argmin}}\ c||r||_2 + \mathcal{J}(\theta, x + r, y), \quad \text{subject to} \quad x + r \in [0, 1]^m \tag{1}$$

This can be interpreted as finding a perturbed image $x + r$ that is closest to x and is misclassified by f. The training procedure for the above framework involves optimizing each layer by using a pool of adversarial examples generated from previous layers. As a training procedure, this is rather cumbersome even when applied to shallow networks having 5-10 layers. To overcome the computational overhead due to the L-BFGS optimization performed at each intermediate layer, [5] propose the Fast Gradient Sign (FGS) method to generate adversarial examples. By linearizing the cost function around the value of the model parameters at a given iteration, they obtain a norm constrained perturbation as follows: $r_{fgs} = \epsilon.sign(\nabla \mathcal{J}(\theta, x, y))$. They show that the perturbed images $x + r_{fgs}$ reliably cause deep models to misclassify their inputs. As noted in [17], the above formulation for adversarial perturbation can be understood by looking at a first order approximation of the loss function $\mathcal{J}(\cdot)$ in the neighborhood of the training sample x:

$$\tilde{\mathcal{J}}(\theta, x + r, y) = \mathcal{J}(\theta, x, y) + \langle \nabla \mathcal{J}(\theta, x, y), r \rangle \tag{2}$$

The FGS solution (r_{fgs}) is the result of maximizing the second term with respect to r, with a l_∞ norm constraint. The central idea behind generating such perturbations using the FGS approach is that training with an adversarial objective function acts as a good regularizer:

$$\tilde{\mathcal{J}}(\theta, x, y) = \alpha \mathcal{J}(\theta, x, y) + (1 - \alpha)\mathcal{J}(\theta, x + r_{fgs}, y) \tag{3}$$

By training the model with both original inputs and adversarially perturbed inputs, the objective function in 3 makes the model more robust to adversaries and provides marginal improvement in performance on the original test data. Intuitively, the FGS procedure can be understood as perturbing each training sample within a L_∞ ball of radius ϵ, in the direction that maximally increases the classification loss.

3.2 Proposed Formulation

In this work, we combine the aspects of the formulations discussed above as follows: (1) Generating adversarial perturbations from intermediate layers rather than just using the input layer (2) Sampling perturbations along directions that moves the training samples towards the neighboring class centers, hence making them harder for the classification task. In order to facilitate the representation of layerwise activations in the loss function, we denote the collection of layerwise responses as $X = \{x_l\}_{l=0}^{L-1}$ and the set of layerwise perturbations as $R = \{r_l\}_{l=0}^{L-1}$. Then, $\mathcal{J}(\theta, X + R, y)$ denotes the loss function where intermediate layer activations are perturbed according to the set R. The notation used in the previous section is a special case where $X = x$ and $R = r_{fgs}$. Now, consider the following objective to obtain the perturbation set R:

$$\underset{R}{\text{argmax}}\ \mathcal{J}(\theta, X + R, y) \quad \text{subject to} \quad ||r_l||_\infty \leq \epsilon,\ \forall l,\ f(X + R) \neq y \tag{4}$$

Ideally, for each training example x, the solution to the above problem, consists of generating the perturbation corresponding to the maximally confusing class; in other words, by choosing the class $\hat{y}$ which maximizes the divergence, $KL(p(y|x_{L-1}), p(\hat{y}|x_{L-1}))$. In the absence of any prior knowledge about class cooccurences, solving this explicitly for each training sample for every iteration is time

consuming. Hence we propose an approximate solution to this problem: the gradients computed from the previous sample at each intermediate layer are cached and used to perturb the activations of the current sample. In a mini-batch setting, this amounts to caching the gradients of the previous mini-batch. To ensure the class constraint in Eq. 4 is satisfied, the only requirement is that successive batches have little lateral overlap in terms of class labels. From our experiments, we observed that any random shuffle of the data satisfies this requirement. Given this procedure of accumulating gradients, we are no longer required to perform an extra gradient descent-ascent step as in the FGS method to generate perturbations for the current batch. Since the gradient accumulation procedure does not add to the computational cost during training, this can be seamlessly integrated into any existing supervised training procedure including even very deep networks as shown in the experiments.

Algorithm 1 Efficient layerwise adversarial training procedure

1: Inputs: Deep network f with loss function $\mathcal{J}$ and parameters θ containing C convolutional blocks. B^t is the batch sampled at iteration t of size k, with input-output pairs $\{X^t, Y^t\}$. Gradient accumulation layers $\{G_c\}_{c=1}^{C}$, with stored perturbations $R^t = \{r_c^t\}_{c=1}^{C}$, initialized with zero. Perturbation parameter, ϵ.
2: t=0:
3: Sample a batch $\{X^t, Y^t\}$ of size k images from the training data
4: Perform regular forward pass - G_c's are not active for $t = 0$.
5: Perform backward pass using the classification loss function. Each gradient accumulation layer G_c stores the gradient signal backpropagated to that layer:

$$r_c^{t+1} = sign(\nabla_c \mathcal{J}(\theta^t, X^t + R^t, Y^t)), \forall c = [1, C] \tag{5}$$

6: **for** t in 1:$|B| - 1$ **do**
7: Sample a batch $\{X^t, Y^t\}$ of size k from the training data
8: Perform forward pass with perturbation: Each gradient accumulation layer acts as follows. Let X_c^t be the input to block c, then:

$$G_c(X_c^t) = X_c^t + \epsilon \cdot r_c^t \tag{6}$$

9: Perform backward pass updating r_c^t to r_c^{t+1} for all blocks c as in Eq. 5 above.
10: **end for**

The training procedure is summarized in Algorithm 1. $sign(\cdot)$ denotes the signum function. We add the gradient accumulation layers after the Batch Normalization layer in each convolutional block (conv-BN-relu). In case BN layers are not present, we add gradient accumulation layers after each convolution layer. A subtle detail that is overlooked in the algorithm is that the value of ϵ is not constant over all the layers, rather it is normalized by multiplying with the range of the gradients generated at the respective layers. During test time, the gradient accumulation layers (G_c) are removed from the trained model.

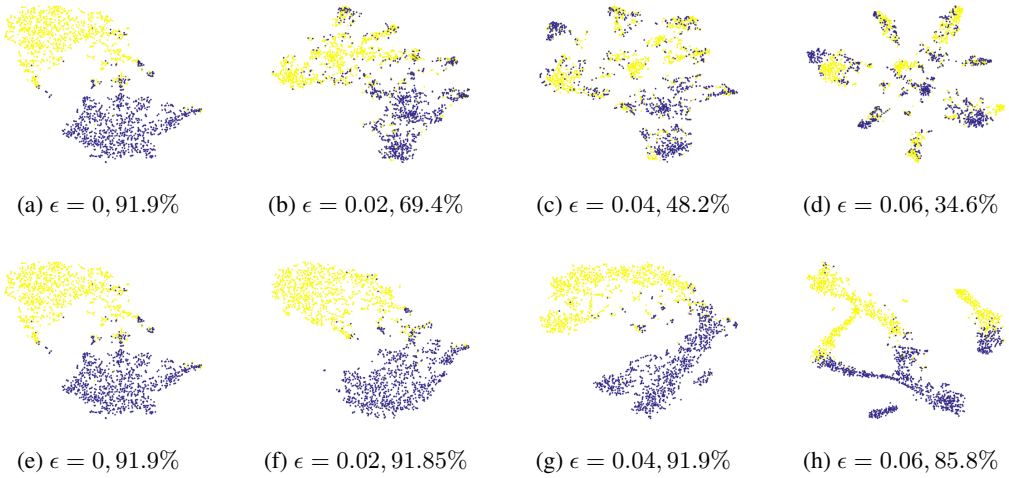
Table 1: Comparison of the strength of adversarial examples between the FGS approach applied at the input and using layerwise perturbations as described in Section 3.2. Reported numbers are classification accuracies for different values of ϵ.

Type	$\epsilon = 0$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.15$	$\epsilon = 0.2$
FGS [5]	92.4	53.28	41.58	36.44	33.85
Ours - all layers	92.4	48.56	21.72	14.76	14.19

3.3 Comparison of adversarial performance

An important question that needs to be addressed in light of the proposed optimization strategy is: Are the gradient directions generated from the previous samples adversarial ? To answer this question, we perform an empirical experiment to measure the performance of a conventionally trained deep model on the test data for CIFAR-10. As described earlier, for each test sample, the intermediate layer activations are perturbed using gradients accumulated from the previous sample. For comparison, we also show the performance of the same model on the adversarial data generated using the FGS method. From the metrics in Table 1, it can be observed that using accumulated gradients from the previous batch as adversarial perturbations results in a bigger drop in performance. This signifies that the aggregated effect of layerwise perturbations is more adversarial compared to perturbing only

the input layer as done in the FGS approach. We performed an additional experiment where only the input layer was perturbed using the gradients of the previous sample instead of perturbing all the intermediate layers. We found that this resulted in negligible drop in the baseline performance, indicating that these gradients are not adversarial enough when used to perturb only the input. Next, we compare the effect of these layerwise adversarial perturbation described in the previous section with random layerwise perturbations. Figure 1 shows a two dimensional t-SNE [11] visualization of the embeddings belonging to the final FC-layer for a range of values of ϵ, the intensity of the adversarial perturbation. We used a pretrained VGG network that was trained on the CIFAR-10



(a) $\epsilon = 0, 91.9\%$ (b) $\epsilon = 0.02, 69.4\%$ (c) $\epsilon = 0.04, 48.2\%$ (d) $\epsilon = 0.06, 34.6\%$

(e) $\epsilon = 0, 91.9\%$ (f) $\epsilon = 0.02, 91.85\%$ (g) $\epsilon = 0.04, 91.9\%$ (h) $\epsilon = 0.06, 85.8\%$

Figure 1: t-SNE visualization of the final fc-layer features of dimension 512 of the VGG network for two randomly chosen classes of the CIFAR-10 data for different values of the intensity, ϵ. The top row shows the adversarial perturbations while the bottom row shows random perturbations of the same intensity. It is clear that the random perturbations do not affect the linear separability of the data, while the adversarial perturbations are extremely effective in leading the network to misclassify the perturbed data.

dataset to compute the embeddings for two randomly chosen classes from the test data. In the bottom row, the effect of random perturbations with zero mean and unit standard deviation, applied layerwise on the original data is also shown. From the visualization and the accuracy values, it can clearly be observed that the layerwise perturbations as described in 3.2 are extremely adversarial to the base network. Notice that even for higher values of ϵ, the data perturbed by layerwise random gradient directions remains clearly linearly separable while the adversarially perturbed data is unable to be distinguished by the base model.

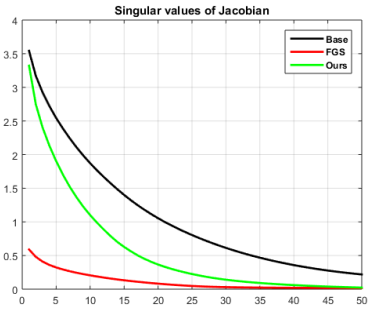


Figure 2: Average singular value (SV) spectrum showing top 50 SVs for the toy example in Section 3.4.

	FGS perturbation from:			
ϵ	Baseline n/w	FGS n/w	Our n/w	
0.0	39.51	39.51	39.51	Baseline n/w
0.02	8.6	32.14	26.92	
0.04	1.92	26.08	18.16	
0.06	0.51	20.74	12.23	
0.08	0.12	16.23	8.63	
0.1	0.03	12.47	6.05	

Table 2: Effect of different perturbations tested on the baseline network for the toy example in Section 3.4.

3.4 Toy example

In order to acquire a better understanding of the regularizing properties of the mapping function learned using the proposed adversarial training approach, we perform a toy experiment using a small neural network consisting of two fully connected layers of sizes 1024 and 512. Each fully connected layer is followed by a hyperbolic tangent activations. We use the grayscale version of the CIFAR-10 dataset as our testbed and L_2 norm weight decay was applied during training. No data augmentation or other regularization methods such as dropout were used during training. We train three networks: a baseline network, a network with the gradient accumulation layers and a network using the FGS training approach described above 3. Cross entropy loss was used to train all the networks. In terms of classification accuracy, the proposed method improves the baseline performance from 39.5% to 43.3% on the original data while the accuracy of the FGS network is 40.5%.

Singular Value Analysis: To gain a deeper understanding of the encoder mapping learned by each network, we perform an analysis similar to [15] by computing the singular values of the Jacobian of the encoder. Since this is a small architecture, we are able to explicitly compute the Jacobian for each sample in the test set. The average singular value spectrum of the Jacobian for the test data is shown in Figure 2. We can make the following observations: (a) The singular value spectrum computed for ours and FGS approach has fewer dominant singular values and decays at a much faster rate compared to base network (b) The FGS training suppresses the response of the network strongly in all the dimensions while our approach achieves a strong suppression only for trailing dimensions. This implies that our network is able to better capture data variations that are relevant for classifying original test data. On the other hand, FGS achieves better robustness against adversarial examples by suppressing network's response strongly even in leading dimensions.

Degree of "adversarialness": In order to validate this implication, we perform an empirical experiment on the CIFAR-10 test dataset. Each test image is used to generate FGS perturbations using each of the three networks trained above. All the three perturbed images are classified by the baseline network. This process is repeated for all 10,000 images from the test set. Table 2 lists the performance of the baseline network on the perturbed images for various values of ϵ, the intensity of the FGS perturbation. Intuitively, this experiment characterizes the relative strengths of adversarial examples generated from the three networks mentioned above. To that end, we can observe that the robustness of the base network to different adversarial examples occur in the following order: FGS $>$ Ours $>>$ base. This concurs with the singular value spectrum analysis presented above that the stronger the suppression of the network's response the less sensitive it becomes to adversarial directions.

3.5 Connection to Robust Optimization

Several regularization problems in machine learning such as ridge regression, lasso or robust SVMs have been shown to be instances of a more general robust optimization framework [19]. To point out the connection between the proposed adversarial training approach and robust optimization, we borrow the idea of uncertainty sets from [17]. To explain briefly, an uncertainty set denoted by $\mathcal{U} = \mathcal{B}_\rho(x, \epsilon)$ represents an epsilon ball around x under norm ρ. [5] point out that adversarial training can be thought of as training with hard examples that strongly resist classification. Under the setting of uncertainty sets, adversarial training with the FGS method could be seen as sampling perturbations from the input space from $\mathcal{U}$ under the l_∞ norm. In this work, we extend the idea of uncertainty sets from input activations to layerwise activations. This can be thought of as sampling perturbations from the feature space learned by the deep network. Let $\mathcal{U}_l$ represent the uncertainty set of the activation x_l at layer l. Then, the proposed adversarial training approach is equivalent to sampling perturbations from the intermediate layer uncertainty sets which makes the feature representation learned at those layers to become more robust during training. Moreover, by generating perturbations from inputs that do not belong to the same class as the current input, the directions sampled from the uncertainty set tend to move the perturbed feature representation towards the direction of an adversarial class. This effect can be observed from the t-SNE visualization shown in Figure 1.

4 Experiments

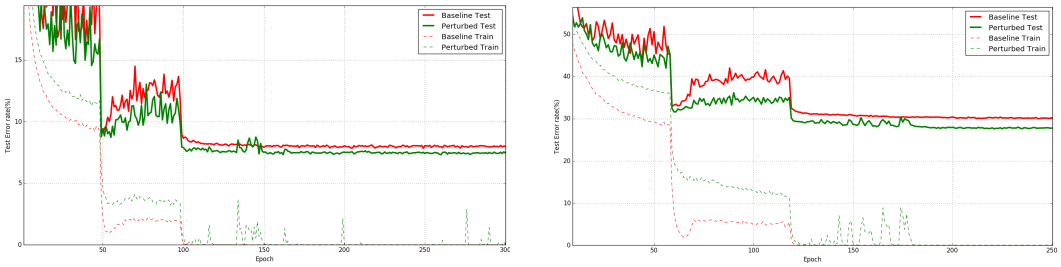
In this section, we provide an experimental analysis of the proposed approach to show that layerwise adversarial training improves the performance of the model on the original test data and increases robustness to adversarial inputs. To demonstrate the generality of our training procedure, we present results on CIFAR-10 and CIFAR-100 [9] using VGG, ResNet-20 and ResNet-56 networks. For the

ResNet networks, we use the publicly available torch implementation [1]. For the VGG architecture, we use a publicly available implementation which consists of Batch Normalization [2]. For all the experiments, we use the SGD solver with Nesterov momentum of 0.9. The base learning rate is 0.1 and it is dropped by 5 every 60 epochs in case of CIFAR-100 and every 50 epochs in case of CIFAR-10. The total training duration is 300 epochs. We employ random flipping as a data augmentation procedure and standard mean/std preprocessing was applied conforming to the original implementations. For the ResNet baseline models, without regularization, we find that they start overfitting if trained longer and hence we perform early stopping and report their best results. For the perturbed models, we find that no early stopping is necessary; the learning continues for a longer duration and shows good convergence behavior. We refer to the model trained using Algorithm 1 as $Perturbed$ throughout this section.

Table 3: Classification accuracy (%) on CIFAR-10 and CIFAR-100 for VGG and Resnet architectures. Results reported are average of 5 runs.

Type	Baseline	Perturbed
VGG	92.1 ± 0.3	92.65 ± 0.2
Resnet-20	90.27 ± 0.4	91.1 ± 0.3
Resnet-56	91.53 ± 0.3	94.1 ± 0.2

Type	Baseline	Perturbed
VGG	69.8 ± 0.5	72.3 ± 0.3
Resnet-20	64.0 ± 0.2	66.9 ± 0.3
Resnet-56	68.2 ± 0.4	71.4 ± 0.5



(a) CIFAR-10 (b) CIFAR-100

Figure 3: Training and test error rates for VGG network trained on CIFAR-10 and CIFAR-100 datasets. The training error rates are computed on the perturbed activations in each epoch.

4.1 Joint loss formulation and comparison with FGS approach

In the proposed training method summarized in Algorithm 1 (referred as Ours-orig), each batch of inputs is perturbed at intermediate layers by the gradients accumulated from the previous batch. In this section, we present an empirical comparison between the following variants:

- FGS-orig: The original FGS joint loss based adversarial training as proposed by [5] and shown in Eq. 3. We used a value of $\alpha = 0.5$; we did not find other values yield any significant improvements.
- FGS-inter: In this setting, different from [5], we use the FGS gradients to perturb the intermediate layer activations and use the joint loss with $\alpha = 0.5$.
- Ours-joint: This setting is same as Ours-orig with the exception that we use the joint loss formulation with $\alpha = 0.5$. Note that, Ours-orig corresponds to setting where $\alpha = 0$

Table 4: Comparison of classification accuracy (%) between various training approaches described in Section 4.1

Type	$\epsilon = 0$	$\epsilon = 0.02$	$\epsilon = 0.04$	$\epsilon = 0.06$	$\epsilon = 0.08$	$\epsilon = 0.1$	Training time
Baseline	89.4	67.5	49.6	41.2	37.3	34.7	x
FGS-orig	88.7	86.4	84.1	81.4	80.5	77.1	2x
FGS-inter	90.9	87.79	83.85	79.65	74.69	69.92	2x
Ours-orig	91.2	87.95	83.84	79.11	73.66	68.37	x
Ours-joint	91.5	86.07	81.38	75.72	70.25	64.76	2x

All the models are trained on the CIFAR-10 dataset. No data augmentation or dropout regularization is applied. The training parameters are similar to the ones used in the previous section. We generate

adversarial test data for the CIFAR-10 test dataset using the FGS method, since it has been shown to generate adversarial examples reliably. We then test the models on the original and adversarial test data for different values of the adversarial strength ϵ. Table 4 shows the results of the different training strategies. $\epsilon = 0$ corresponds to the original test data and other values of ϵ indicate the strength of adversarial FGS perturbation added to the input image. From these results, we make the following observations: (1) Approaches based on perturbing intermediate layers (FGS-inter,Ours-orig,Ours-joint) improve the performance on the original data significantly as compared to perturbing only the input but they marginally decrease the adversarial test performance. (2) On the other hand, perturbing only the input layer (FGS-orig) yields the best adversarial test performance among the compared approaches while performing marginally worse than the baseline on the original test data. These observations indicate the possibility of a trade-off that exists between adversarial robustness and classification performance.

4.2 Comparison with Dropout

We perform an experiment where we compare the regularization performance of the proposed adversarial training to Dropout. We use the VGG architecture used in the previous sections and perform experiments with and without dropout on CIFAR-10 and CIFAR-100 datasets. To understand the full extent of the regularization performance, we did not perform any data augmentation for this experiment.

Table 5: Comparison of classification accuracy (%) with/without dropout on CIFAR-10 and CIFAR-100 for the VGG model

Type	Baseline	Perturbed
w/o Dropout	89.4	91.3
with Dropout	91.5	92.1

Type	Baseline	Perturbed
w/o Dropout	69.8	72.3
with Dropout	70.5	73.1

The following observations could be made from Table 5: (1) The perturbed model performs better than the baseline model with or without dropout. Thus, the proposed training improves the performance of even dropout based networks. (2) On a complex task like CIFAR-100, the proposed adversarial training approach gives better regularization performance compared to that provided by dropout (70.5% (vs) 72.3%). Since the proposed adversarial perturbations are intended to move the inputs towards directions that strongly resist correct classification, they are able to create a more discriminative representation for tasks with a larger number of classes.

Table 6: Effect of layerwise adversarial perturbations on the classification accuracy using the VGG network on the CIFAR-10 dataset. Baseline performance is 89.4%

Layer (conv1 to)	pool1	pool2	pool3	pool4	pool5
Accuracy	89.5	89.62	90.24	91.1	91.3

4.3 Perturbing deeper layers

In this section, we analyze the effect of adversarial perturbations starting from the lowest convolutional layers which model edges/shape information to the more deeper layers which model abstract concepts. For this experiment, we use the VGG network with batch normalization that was used in the previous section. The experiments were performed on the CIFAR-10 dataset. No data augmentation or dropout is applied. It is clear from the results in Table 6 that the improvement in performance due to the proposed layerwise perturbations become significant when applied to the deeper layers of the network, which is in line with the observation made by [21]. While performing layerwise alternate training as proposed by[21] becomes infeasible for even moderately deep architectures, our training scheme provides an efficient framework to infuse adversarial perturbations throughout the structure of very deep models.

5 Summary and Conclusion

While the behavior of CNNs to adversarial data has generated some intrigue in computer vision since the work of [21], its effects on deeper networks have not been explored well. We observe that adversarial perturbations for hidden layer activations generalize across different samples and we leverage this observation to devise an efficient adversarial training approach that could be used

to train very deep architectures. Through our experiments and analysis we make the following observations: (1) Contrary to recent methods which are inconclusive about the role of perturbing intermediate layers of a DNN in adversarial training, we have shown that for very deep networks, they play a significant role in providing a strong regularization (2) The aggregated adversarial effect of perturbing intermediate layer activations is much stronger than perturbing only the input (3) Significant improvement in classification accuracy entails capturing more variations in the data distribution while adversarial robustness can be improved by suppressing the unnecessary variations learned by the network 3.4. By providing an efficient adversarial training approach that could be used with very deep models, we hope that this can inspire more robust network designs in the future.

Appendix

Results on Wide Residual Networks (WRN)

Wide Residual Networks are recently proposed deep architectures that generated state of the art results on CIFAR-10 and CIFAR-100 datasets. In this experiment, we use their publicly available implementation and train them from scratch using the proposed adversarial training approach using the parameter settings described in the original paper in Section 4. Specifically, the Ours-joint approach described in Section 4.1 is used for training. As data augmentation, we applied flipping and random cropping as done in their native implementation. The results are shown in Table 7.

Table 7: Classification error rates (%) on CIFAR-10 and CIFAR-100 for WideResNet (WRN) architectures. Our results are reported as average of 5 runs. For comparison we provide the published WRN baseline results. $^{(*)}$ denotes the results obtained by a single run.

Model	#params	CIFAR-10	CIFAR-100
WRN-28-10	36.5M	4.00	19.25
WRN-28-10 with dropout	36.5M	3.89	18.85
WRN-40-10 with dropout*	51.0M	3.8	18.3
WRN-28-10 with Ours-joint	36.5M	**3.62** $\pm$ 0.05	**17.1** $\pm$ 0.1

Response to local perturbation depends on the Jacobian

Let $f : \mathbb{R}^m \mapsto \mathbb{R}^n$ be a mapping between two metric spaces (euclidean, for simplicity) . Then, for $x \in \mathbb{R}^m$, let $J_f(x) \in \mathbb{R}^n \times \mathbb{R}^m$ denote the jacobian of f evaluated at x. Let $\delta x \in \mathbb{R}^m$ be a bounded local perturbation in the neighborhood of x. A first order truncated expansion of $f(x + \delta x)$ is given as:

$$f(x + \delta x) = f(x) + J_f(x)^T \delta x$$

We can bound the frobenius norm of the second term as follows:

$$||J_f(x)^T r||_F \overset{(a)}{\leq} ||J_f(x)||_F \; ||\delta x||_2 = \sqrt{\sum_{i=1}^{min(m,n)} \sigma_i^2} \cdot ||\delta x||_2$$

$$\implies ||J_f(x)^T \delta x||_F \leq \sqrt{\sum_{i=1}^{min(m,n)} \sigma_i^2} \cdot ||\delta x||_2$$

where $||\cdot||_F$ denotes the frobenius norm; for vectors, it is the same as the L_2 norm and σ_i denotes the i^{th} singular value of the Jacobian; (a) is a direct application of Cauchy-Schwarz inequality. Applying this result to the singular value spectra plotted in Figure 2 in the paper, we see that the base network without adversarial training is extremely sensitive to local perturbations compared to adversarially trained networks using FGS and the proposed approach.

Setting ϵ parameter

Following the notation from Section 3, let $\nabla_l \mathcal{J}(\theta, x, y)$ denote the gradient of the loss function backpropagated to the l^{th} layer. Let $M = max(\nabla_l \mathcal{J}(\theta, x, y))$, $m = min(\nabla_l \mathcal{J}(\theta, x, y))$. Then, the value of ϵ for each layer is calculated as: $\epsilon_l = \epsilon \cdot (M - m) \; \forall l$, where $\epsilon \in \{10, 20, 30\}$. The exact value is cross-validated using a held out set. In practice, we found our training approach to not be

overly sensitive to ϵ. We tuned ϵ only for the VGG network on CIFAR-10 and used the same value for all the other networks such as ResNets and WideResNets on both CIFAR-10 and CIFAR-100 datasets. Note that, for cases where a fixed value of ϵ is specified such as in Figure 1 in the paper, the same value is used for all layers ignoring the normalizing factor, $(M - m)$.

Acknowledgement

Swami Sankaranarayanan and Rama Chellappa are supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA R&D Contract No. 2014-14071600012. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

References

[1] Resnet torch implementation, 2017. URL `https://github.com/facebook/fb.resnet.torch`.

[2] Vgg torch implementation with batch normalization, 2017. URL `https://github.com/szagoruyko/wide-residual-networks`.

[3] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *arXiv preprint arXiv:1502.02590*, 2015.

[4] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. *CoRR*, abs/1608.08967, 2016. URL `http://arxiv.org/abs/1608.08967`.

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

[6] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[9] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

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

[11] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[12] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.

[13] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[14] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE, 2016.

[15] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 833–840, 2011.

[16] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[17] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. *arXiv preprint arXiv:1511.05432*, 2015.

[18] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[19] Suvrit Sra, Sebastian Nowozin, and Stephen J Wright. *Optimization for machine learning*. Mit Press, 2012.

[20] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[21] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[22] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L. Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In Sanjoy Dasgupta and David Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 1058–1066. JMLR Workshop and Conference Proceedings, May 2013. URL `http://jmlr.org/proceedings/papers/v28/wan13.pdf`.

[23] Beilun Wang, Ji Gao, and Yanjun Qi. A theoretical framework for robustness of (deep) classifiers under adversarial noise. *CoRR*, abs/1612.00334, 2016. URL `http://arxiv.org/abs/1612.00334`.

[24] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016. URL `http://arxiv.org/abs/1605.07146`.

[25] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4480–4488, 2016.