Early Layers Are More Important For Adversarial Robustness

Can Bakiskan [1] **Metehan Cekic** [1] **Upamanyu Madhow** [1]

Abstract

Adversarial training and its variants have become the de facto standard for combatting against adversarial attacks in machine learning models. In this paper, we seek insight into how an adversarially trained deep neural network (DNN) differs from its naturally trained counterpart, focusing on the role of different layers in the network. To this end, we develop a novel method to measure and attribute adversarial effectiveness to each layer, based on partial adversarial training. We find that, while all layers in an adversarially trained network contribute to robustness, earlier layers play a more crucial role. These conclusions are corroborated by a method of tracking the impact of adversarial perturbations as they flow across the network layers, based on the statistics of "perturbation-to-signal ratios" across layers. While adversarial training results in black box DNNs which can only provide empirical assurances of robustness, our findings imply that the search for architectural principles in training and inference for building in robustness in an interpretable manner could start with the early layers of a DNN.

1. Introduction

Since the vulnerability of neural networks to adversarial examples was first pointed out (Biggio et al., 2013; Szegedy et al., 2014), there has been significant research effort in developing more sophisticated attacks and defenses. Most defenses which attempt to employ structural insights to guide their design have been beaten or circumvented (Carlini & Wagner, 2017b;a; Athalye et al., 2018). What remains standing is adversarial training (Madry et al., 2018) and other defenses that incorporate it (Gowal et al., 2021; Zhang et al., 2019). Adversarial training is essentially a "black box" technique in which adversarially perturbed examples are generated and used for training as the network parameters are evolving. In view of its success, it is naturally under the spotlight for researchers looking to improve state-of-the-art and/or looking for an answer to how the learned model differs from a model trained in a standard fashion. Previous works in the literature attempt to analyze adversarial training in terms of loss landscape (Liu et al., 2020), decision boundary (He et al., 2018), class-wise robustness (Tian et al., 2021), smoothness (Kanai et al., 2021) and algorithmic stability (Xing et al., 2021). However, prior work does not provide structural insight into the contribution of different layers in the network towards robustness.

In this paper, we seek to quantify the importance of each layer in handling adversarial perturbations, comparing adversarial training versus natural training on a layerwise basis. Our goal is to develop architectural insights to guide further research into the structural properties of robust neural networks. We demonstrate that earlier layers play a crucial role in defending against adversarial perturbations. We also provide a novel approach for inspecting how adversarial perturbations flow through the layers of the network, thus providing a fine-grained analysis of a model's robustness. We report experiments on CIFAR-10 image classification using two popular architectures, namely VGG and ResNet.

Our contributions are:

- We develop a partial adversarial training method in order to quantify the role of each layer in providing robustness against adversarial attacks.
- We introduce the concept of "perturbation-to-signal ratio" (PSR) defined in terms of different ℓ_p norms, to track the flow of adversarial perturbations through the network.
- Using these two techniques we conclude that earlier layers play a crucial role in adversarial robustness.

2. Background and Related Work

The discovery of adversarial examples started a cat and mouse game between attackers and defenders. The fundamental importance of this contest is well understood by the research community: given the pervasive impact of DNNs,

[1]Department of Electrical Engineering, UC Santa Barbara, Santa Barbara, CA, USA. Correspondence to: Can Bakiskan <canbakiskan@ucsb.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

it is crucial to understand and address their vulnerabilities. While attackers try to identify the smallest potential distortion (adversarial attack) that can fool a DNN (flipping the prediction in a classification context), defenders aim to defend DNNs against such threats.

2.1. Attacks

By definition, adversarial attacks are constrained not to change ground-truth labels, and are commonly bounded by ℓ_p norms. In other words, while successfully fooling the DNNs, they are restricted to have no effect on the decision made by humans. Although there are sufficiently successful adversarial attacks that do not need access to the inner structure of the models (black box attacks) (Papernot et al., 2017; Chen et al., 2017; Andriushchenko et al., 2020), the most powerful adversarial attacks utilize access to the full knowledge of the model (white box attacks) (Goodfellow et al., 2014; Kurakin et al., 2016). State of the art whitebox attacks are based on gradient ascent on the cost function, compute the perturbation using normalized gradients with respect to the inputs. These include the fast gradient sign method (FGSM) (Goodfellow et al., 2015), the iterative version of FGSM known as the basic iterative method (BIM) (Kurakin et al., 2016), and their derivatives (Madry et al., 2018; Croce & Hein, 2020), are often referred to as projected gradient descent (PGD). For this study, we work with the PGD attack, which is regarded as one of the most effective first order ℓ^∞ bounded adversarial attacks.

PGD computes the perturbation iteratively as follows:

$$\boldsymbol{e}_{i+1} = \text{clip}_\epsilon\big[\boldsymbol{e}_i + \delta \cdot \text{sign}(\nabla_{\mathbf{e}}\mathcal{L}(\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{e}_i), \boldsymbol{y}))\big] \quad (1)$$

where $\mathbf{e}_i$ corresponds to the value of the perturbation at iteration i with $\mathbf{e}_0 = \mathbf{0}$ or $\mathbf{e}_0$ with each element drawn from uniform distribution $\mathcal{U}(-\epsilon, \epsilon)$, ϵ is the overall ℓ^∞ attack budget, and δ is the step size for each iteration.

2.2. Defenses

Plenty of sophisticated defense mechanisms have been proposed employing detection techniques (Feinman et al., 2017; Xu et al., 2017; Grosse et al., 2017; Pang et al., 2018), preprocessing methods (Guo et al., 2017; Bhagoji et al., 2018), modification of the optimization objective (Hein & Andriushchenko, 2017; Xu et al., 2017; Moosavi-Dezfooli et al., 2019), and biological constraints (Marblestone et al., 2016; Nayebi & Ganguli, 2017; Dapello et al., 2020; Cekic et al., 2022). However, these methods have either eventually been beaten by an adaptive attack (Carlini & Wagner, 2017b;a; Athalye et al., 2018) or do not match the empirical success of adversarial training. Adversarial training and its variants therefore continue to be the state of the art defense against adversarial attacks.

2.2.1. Adversarial Training and Variants

In adversarial training, first proposed in (Madry et al., 2018), adversarial perturbations are added during the training phase to increase robustness. Since then, most successful defense methods are either variants of adversarial training (Dong et al., 2018; Zhang et al., 2019; Wang et al., 2019; Rice et al., 2020), or they incorporate adversarial training to improve robustness (Dai et al., 2021; Sridhar et al., 2021; Kang et al., 2021). Despite the success of adversarial training, there remains a large generalization gap. Schmidt et al. (2018) articulated the need for additional data to close this gap, which inspired many notable works that generate additional training data (Rebuffi et al., 2021) and enlarge the datasets with new augmentation techniques (Gowal et al., 2021). As mentioned, while siginificant effort has been devoted to analysis of adversarial training (Liu et al., 2020),(He et al., 2018),(Tian et al., 2021),(Kanai et al., 2021),(Xing et al., 2021), but to the best of our knowledge, there has been no systematic study of how different layers in the DNN contribute to the robustness of the model. This is the gap that we aim to fill with this work, as a first step to obtaining detailed structural insight into what makes a DNN robust.

3. Setup and Definitions

We now describe the methodology we use to evaluate and understand the importance of each layer.

3.1. Partial Adversarial Training

Since our goal is to analyze the role of each layer of the network in adversarial training, we found the most straighforward method to be restricting the contribution of each layer to the adversarial training process. We achieve this by a two step process (see Figure 1). In the first training stage, we train the whole network adversarially or naturally. Then, we freeze either the earlier or later part of the net-

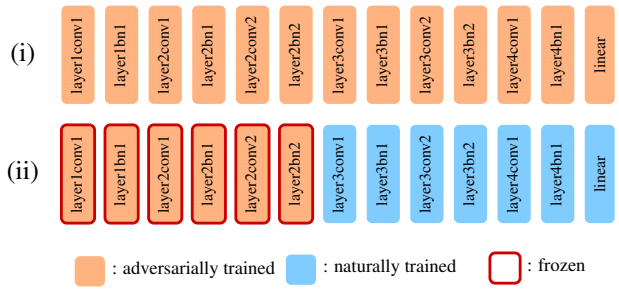

Figure 1: Visualization of (i) initial training, (ii) freezing and reinitializing and retraining for an example scenario corresponding to A1N2 (all layers trained adversarially first; then earlier layers are frozen and later layers are reinitialized and trained naturally) with cutoff before layer 3 convolutional 1. Note that skip connections have been omitted and network is shortened for illustration purposes.

work and reinitialize the remaining unfrozen layers. In the second training stage, we train the unfrozen layers adversarially or naturally from scratch. To keep track of and refer to various experiments of these two kinds, we employ a 4 character code. The first two characters of the code refer to how the earlier layers are trained and whether trained in first or second stage; and the last two characters refer to how the later layers are trained and in which stage. "A" means adversarial, "N" means natural training. "1" means that part of the network was trained first, then frozen, "2" means that part of the network was reinitialized and retrained a second time. Putting it all together, an experiment with the code "A1N2" is where we train all layers in the network adversarially first, then freeze the earlier layers and reinitialize later layers, and retrain later layers naturally. An experiment with the code "A2A1" means we train all the layers in the network adversarially first. Following that, we freeze the later layers and reinitialize earlier layers. Then, we retrain earlier layers adversarially once again. We use this coding scheme to refer to specific instances of partial adversarial training throughout the paper.

To remove the confounding effects caused by the reinitialization of trainable parameters, non-trainable statistics, and the mismatch of the optimizer state from the effect of partial adversarial training, we retrain the reinitialized layers both adversarially and naturally, and report the differences between them.

In order to reduce the training time of the substantial number of different experiments, we opted for a smaller ResNet (He et al., 2015) and VGG (Simonyan & Zisserman, 2014) models with 16 units. Number of parameters is 11.2 million for ResNet and 14.7 million VGG.

We should mention that since ResNet has skip connections, it isn't entirely linear. However, in order to plot the difference in adversarial losses in a two dimensional plot, we need to use a linear list of layers. While linearizing the layers of ResNet, we chose to place shortcut layers after the main branch layers that it's summed with, but the placement of these shorcut layers are somewhat arbitrary.

Another important point to make here is that the distributional shift in layer inputs caused by adversarial training necessitates the running mean and running variance (non-trainable statistics) of batch norm layers be kept unfrozen after the reinitialization. To be specific, freezing and reinitializing works in a straighforward way for trainable parameters (both convolutional and batch norm layers). As for non-trainable statistics, if they belong to a "frozen" batch norm layer (whose trainable parameters are frozen) they themselves are not frozen, but not reinitialized either. If they belong to a batch norm layer that isn't frozen, then they are reinitialized. This is done so because freezing the statistics would significantly hamper the training due to the different distributions of batch norm layer inputs between when clean and adversarial images are fed to the network.

3.1.1. Retraining Later Layers

The first type of partial training experiment we conduct is where we freeze the earlier layers, then reinitialize and retrain the later layers. This type of training is the most natural of the three partial adversarial training setups because it resembles the regular procedure followed in transfer learning.

3.1.2. Retraining Earlier Layers

The second type of partial training experiment is where we freeze the later layers, then reinitialize and retrain the earlier layers.

3.1.3. Retraining A Single Layer

Similar to the previous two techniques, one can also freeze all but one layer, and then reinitialize and retrain that single layer. This makes a small but noticeable change in the behavior of the neural network.

3.2. Tracking Perturbations Across Layers

We also use another method to visualize how different layers operate on the adversarial perturbations by plotting the ℓ_p norm of the adversarial layer outputs and clean layer outputs. We divide this by the ℓ_p norm of the clean layer outputs to obtain perturbation-to-signal-ratio (Eq 2). This serves two purposes: we get a better measure of the strength of the attack at that level with respect to the clean layer outputs, and the effect of number of dimensions on the ℓ_p norm is eliminated. While ℓ_p norms don't capture the full complexity of the adversarial perturbations, they do equip us with an approximate measure of remaining power of the perturbation at that layer.

$$\text{PSR}_l = \frac{\|\boldsymbol{f}_l(\boldsymbol{x}) - \boldsymbol{f}_l(\boldsymbol{x}+\boldsymbol{e})\|_p}{\|\boldsymbol{f}_l(\boldsymbol{x})\|_p} \tag{2}$$

4. Experiments & Results

For completeness, we train all combinations of partial adversarial training (A1N2, N1A2, A1A2, N1N2, A2N1, N2A1, A2A1, N2N1) and compute the differences of the statistics between each combination that differ in a single letter (e.g. A1N2 and N1N2 or A1N2 and A1A2). The extended results table can be found in the Appendix for all partial adversarial training combinations.

At first glance, accuracy might seem like a reasonable statistics to measure effectiveness of layers in adversarial training. However, when we look at the plots for the accuracy differences (Figures 2a and 2b) we notice a shortcoming of

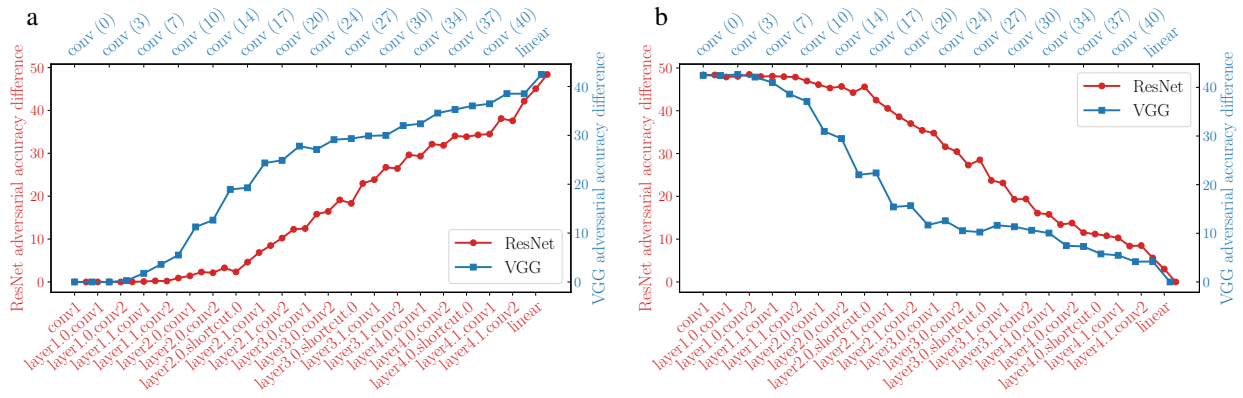

Figure 2: The difference in adversarial **accuracy** between **(a)** all adversarial and partially adversarial networks (|A1A2-A1N2|) and **(b)** partially adversarial and all natural networks (|A1N2-N1N2|) as the number of adversarially trained layers increases. Note the plateau at the beginning.

using accuracy as a performance measure. Namely, both of the plots show saturation of accuracy differences at the beginning. The saturation at the beginning means that even if the first quarter of the network (for ResNet) is adversarially trained, there is no observed gain in robustness. This is somewhat counterintuitive, and indeed, when we look at the same plots for cross entropy loss, which is what the adversarial attack tries to maximize in the first place, we see a different picture. The first layers certainly do increase robustness, in the form of reduced false-confidence in the wrong label. To give a concrete example, for a hypothetical two label task, both $[0.51, 0.49]$ and $[0.99, 0.01]$ softmax outputs give 0% accuracy against label of $[0, 1]$. However, clearly the second softmax output is much worse. Therefore, when the accuracies observed are close to zero, the better approach to measure effectiveness is to use the cross entropy loss itself. If, on the other hand, accuracies are not close to zero, then accuracy can also be a viable metric.

4.1. Details

To reduce the amount of hyperparameter search and hand-tailoring, and for consistency, we use the same optimizer, scheduler and optimization hyperparameters for all experiments of the same model. We train ResNet-16 with momentum SGD optimizer with learning rate 0.1 for 75 epochs and 0.01 for a further 5 epochs. We use a momentum coefficient of 0.9, a weight decay coefficient of 2×10^{-4} and batch size of 128. We start training VGG-16 with Adam optimizer with learning rate 1×10^{-3} for 100 epochs. We scale down the learning rate by a factor of 10 at epochs 50 and 75. We don't use weight decay for VGG and use a batch size of 128. The choice of using different optimizers for different models was made to ensure diversity and reduce the likelihood of observations being caused by the choice a certain optimizer/hyperparameter. The hyperparameters for the SGD optimizer were taken from (Madry et al., 2018) and the hyperparameters for Adam were determined after a hyperparameter search in a limited number of combinations. For ResNet training we stop early after reducing the learn-

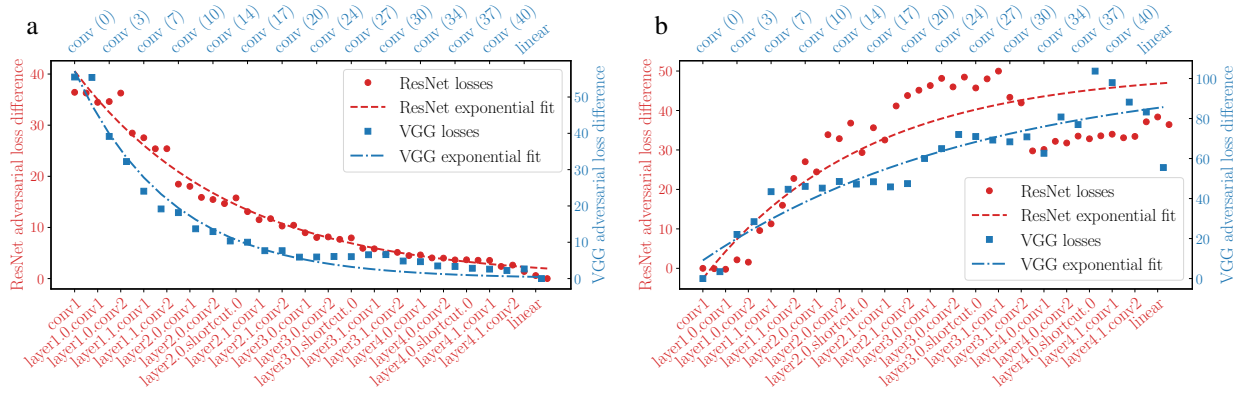

Figure 3: The difference in adversarial **loss** between **(a)** partially and all adversarial networks (|A1N2-A1A2|) and **(b)** all natural and partially adversarial networks (|N1N2-A1N2| as the number of adversarially trained layers increases.

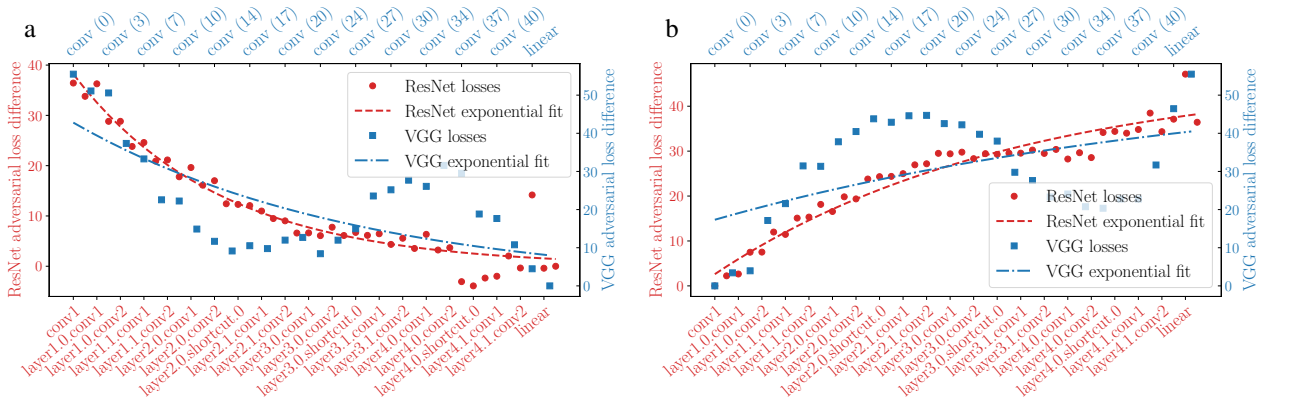

Figure 4: The difference in adversarial **loss** between **(a)** all natural and partially adversarial networks (i.e. |N2N1-N2A1|) and **(b)** partially and all adversarial networks (|N2A1-A2A1|) as the number of adversarially trained layers decreases.

ing rate to reduce the chance of catasrophic overfitting (Rice et al., 2020). We don't use this for VGG, to reduce the relience of observations on a particular method, as mentioned before. Following (Madry et al., 2018), we use ℓ_∞ bounded PGD attack with attack budget $\epsilon = 8/255$, step size $2/255$, and number of steps 10 for both of the models' adversarial training. Perturbations are initialized with each element drawn from uniform distribution $\mathcal{U}(-\epsilon, \epsilon)$. For adversarial testing, we use ℓ_∞ bounded PGD attack with attack budget $\epsilon = 8/255$, step size $1/255$, and number of steps 100. Perturbations are initialized with each element drawn from uniform distribution $\mathcal{U}(-\epsilon, \epsilon)$.

4.2. Retraining Later Layers

We observe that in both Figure 3a and Figure 3b as the number of adversarially trained layers increases from zero to number of layers, the adversarial loss goes from all natural model's loss value to all adversarial model's loss value as expected. What is striking is how this transition occurs. The evolution of adversarial loss difference very closely follows an exponential function for both of the models. This indicates that the first few layers play a major role in reducing the adversarial loss.

Another key observation we made is that if the earlier layers are trained naturally in the first stage, frozen and the rest is adversarially trained (N1A2), the network has a more difficult time converging. In fact, after some cutoff point (layer3.0.bn2 for ResNet and layer 11 for VGG) the network cannot learn and collapses to random guessing. We think this is due to the perturbation growing to such high levels in the earlier layers that the distributions seen by the later layers are not learnable.

4.3. Retraining Earlier Layers

In Figures 4a and 4b, similar to what we observe in later layer retraining, we see a change in adversarial loss from all adversarial model's loss value to all natural model's loss value as the number of adversarially trained layers decreases. The transition for ResNet again follows an exponential curve wheres the the transition for VGG follows a wavy looking curve. We are not sure exactly what causes this behavior and why it is only observed when earlier layers are retrained.

4.4. Retraining A Single Layer

When we freeze the whole network and retrain a single layer in Figure 5, we see a similar trend in the statistics as in Sections 4.2 and 4.3. Specifically, the change in accuracy induced by each individual early layer is much greater than the change induced by later layers. Interestingly, the effect of individual layers follow a more linear trend. In this experiment, we can safely report the accuracies since changing only a single layer does not bring the adversarial accuracy close to zero where the aforementioned saturation occurs.

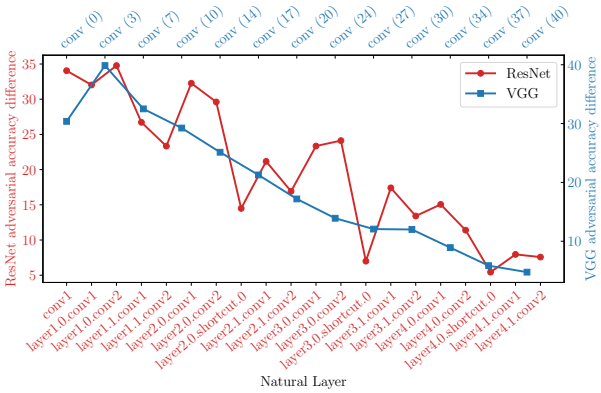

Figure 5: The difference in adversarial **accuracy** between all adversarial and network with a single natural layer. Indicating the singular contribution of that layer to adversarial training.

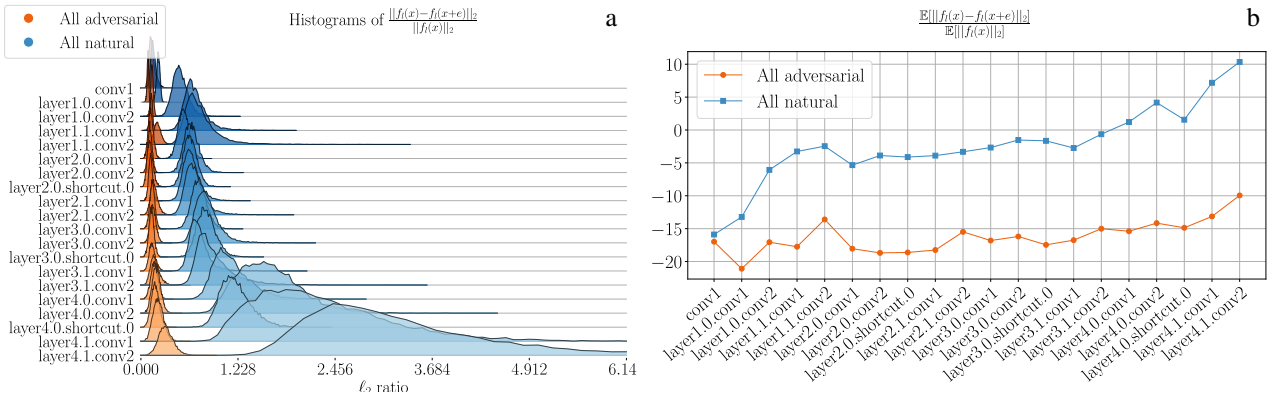

Figure 6: **(a)** Distributions of the ratio of adversarial perturbations' ℓ_2 norm to signal's ℓ_2 norm after each convolutional layer. Histogram is computed by computing this value for each image. **(b)** Means of the distributions of PSR in part a in dB.

4.5. Tracking Perturbations Across Layers

Using the perturbation-to-signal ratio tracking (Eq. 2) we track and plot the sustained power of the adversarial perturbation at each layer. Figure 6a shows this ratio computed for each image and the resulting ratio distributions. Figure 6b shows the means of these distributions converted to dB. This value is, in essence, the negative of the signal-to-noise ratio (SNR dB) widely used in the signal processing field, if we view the perturbation as noise. We can see that the difference in PSR grows rapidly in the earlier layers, and continues to grow in the later layers, albeit at a slower rate.

5. Discussion

Our partial adversarial training approach clearly reveals that earlier layers play a bigger role in providing robustness. This is demonstrated both by our evaluations of accuracy under adversarial attack and by the trends in the loss function. An intuitive explanation is that, unless adversarial perturbations are attenuated early on, they can blow up as they flow through the network. This intuition is borne out by our comparisons of perturbation-to-signal ratio between adversarially trained and naturally trained networks: we see from Figure 6a how the perturbation blows up as it flows through the layers in the naturally trained network, and how it is kept under check with adversarial training.

Yet another piece of evidence regarding the importance of the early layers is an experiment in which we freeze the early layers after *natural* training, and then try to train later layers adversarially. We find that, if enough early layers are frozen after natural training, adversarial retraining simply does not converge: this is because the perturbation is allowed to blow up too much as it flows through the naturally trained layers.

Limitations: Due to the sheer number of experiments to be run, we could not run them more than once and we could not report error bars associated with the experimental results. Therefore there is some statistical variation which results in minor roughness in graphs. It also means that if the experiments were conducted again, slightly different results would be obtained. However, since the adjustment of cutoff points is very granular (one layer at a time), this statistical variation does not significantly affect the conclusions drawn.

6. Conclusion

We hope that our observations, and the methods we have developed to obtain them, open the door to further efforts at dissecting adversarial training, and in developing principled approaches to designing in robustness. For example, it may be possible to be more aggressive about adversarial training in the early layers as we continue to seek methods which trade off clean and attacked accuracy. As another example, while defenses based on pre-processing input data have been defeated, a closer analysis of the early layers may provide insights for the design of improved preprocessing techniques. In terms of measuring robustness, quick computations of perturbation-to-signal ratios for early layers may be more informative about the operation of a layer than exhaustive evaluations of end-to-end accuracy.

An open issue is whether the structure of the weights, and the patterns of the activations, is different for adversarially trained networks than for natural trained networks. While we have performed extensive experiments to gain insight into such questions, we were unable to extract clear guidelines that apply to the different network architectures we have considered. and therefore do not report them here. However, we believe that continued efforts to gain detailed structural insight are of great value in the quest to develop principled, interpretable approaches to robustness.

Acknowledgment

This work was supported in part by the Army Research Office under grant W911NF-19-1-0053, and by the National Science Foundation under grants CIF-1909320 and CIF-2224263.

References

Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pp. 484–501. Springer, 2020.

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

Bhagoji, A. N., Cullina, D., Sitawarin, C., and Mittal, P. Enhancing robustness of machine learning systems via data transformations. In *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–5. IEEE, 2018.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 387–402. Springer, 2013.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pp. 39–57, 2017a.

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

Cekic, M., Bakiskan, C., and Madhow, U. Neuro-inspired deep neural networks with sparse, strong activations. *arXiv preprint arXiv:2202.13074*, 2022.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pp. 15–26, 2017.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2206–2216. PMLR, 13–18 Jul 2020.

Dai, S., Mahloujifar, S., and Mittal, P. Parameterizing activation functions for adversarial robustness. *arXiv preprint arXiv:2110.05626*, 2021.

Dapello, J., Marques, T., Schrimpf, M., Geiger, F., Cox, D., and DiCarlo, J. J. Simulating a primary visual cortex at the front of cnns improves robustness to image perturbations. *Advances in Neural Information Processing Systems*, 33:13073–13087, 2020.

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., and Li, J. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185–9193, 2018.

Feinman, R., Curtin, R. R., Shintre, S., and Gardner, A. B. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

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

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

Gowal, S., Rebuffi, S.-A., Wiles, O., Stimberg, F., Calian, D., Mann, T., and DeepMind, L. Doing more with less: Improving robustness using generated data. *ICLR Workshop on Security and Safety in Machine Learning Systems*, 2021.

Grosse, K., Manoharan, P., Papernot, N., Backes, M., and McDaniel, P. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

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

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, W., Li, B., and Song, D. Decision boundary analysis of adversarial examples. In *International Conference on Learning Representations*, 2018.

Hein, M. and Andriushchenko, M. Formal guarantees on the robustness of a classifier against adversarial manipulation. *Advances in neural information processing systems*, 30, 2017.

Kanai, S., Yamada, M., Takahashi, H., Yamanaka, Y., and Ida, Y. Smoothness analysis of adversarial training. *arXiv preprint arXiv:2103.01400*, 2021.

Kang, Q., Song, Y., Ding, Q., and Tay, W. P. Stable neural ode with lyapunov-stable equilibrium points for defending against adversarial attacks. *Advances in Neural Information Processing Systems*, 34, 2021.

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

Liu, C., Salzmann, M., Lin, T., Tomioka, R., and Süsstrunk, S. On the loss landscape of adversarial training: Identifying challenges and how to overcome them. *Advances in Neural Information Processing Systems*, 33:21476–21487, 2020.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

Marblestone, A. H., Wayne, G., and Kording, K. P. Toward an integration of deep learning and neuroscience. *Frontiers in computational neuroscience*, pp. 94, 2016.

Moosavi-Dezfooli, S.-M., Fawzi, A., Uesato, J., and Frossard, P. Robustness via curvature regularization, and vice versa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9078–9086, 2019.

Nayebi, A. and Ganguli, S. Biologically inspired protection of deep networks from adversarial attacks. *arXiv preprint arXiv:1703.09202*, 2017.

Pang, T., Du, C., Dong, Y., and Zhu, J. Towards robust detection of adversarial examples. *Advances in Neural Information Processing Systems*, 31, 2018.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pp. 506–519, 2017.

Rebuffi, S.-A., Gowal, S., Calian, D. A., Stimberg, F., Wiles, O., and Mann, T. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946*, 2021.

Rice, L., Wong, E., and Kolter, Z. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pp. 8093–8104. PMLR, 2020.

Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Madry, A. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*, pp. 5014–5026, 2018.

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

Sridhar, K., Sokolsky, O., Lee, I., and Weimer, J. Improving neural network robustness via persistency of excitation. *arXiv preprint arXiv:2106.02078*, 2021.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Tian, Q., Kuang, K., Jiang, K., Wu, F., and Wang, Y. Analysis and applications of class-wise robustness in adversarial training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1561–1570, 2021.

Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., and Gu, Q. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2019.

Xing, Y., Song, Q., and Cheng, G. On the algorithmic stability of adversarial training. *Advances in Neural Information Processing Systems*, 34, 2021.

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

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pp. 7472–7482. PMLR, 2019.

A. Appendix

A.1. Code

The source code for the paper can be found in the supplementary materials `zip` file or at:
https://anonymous.4open.science/r/layersnotequalinAT-1/

A.2. Network Architectures

We depict below, our layer numbering schemes for the ResNet and VGG networks used in our experiments.

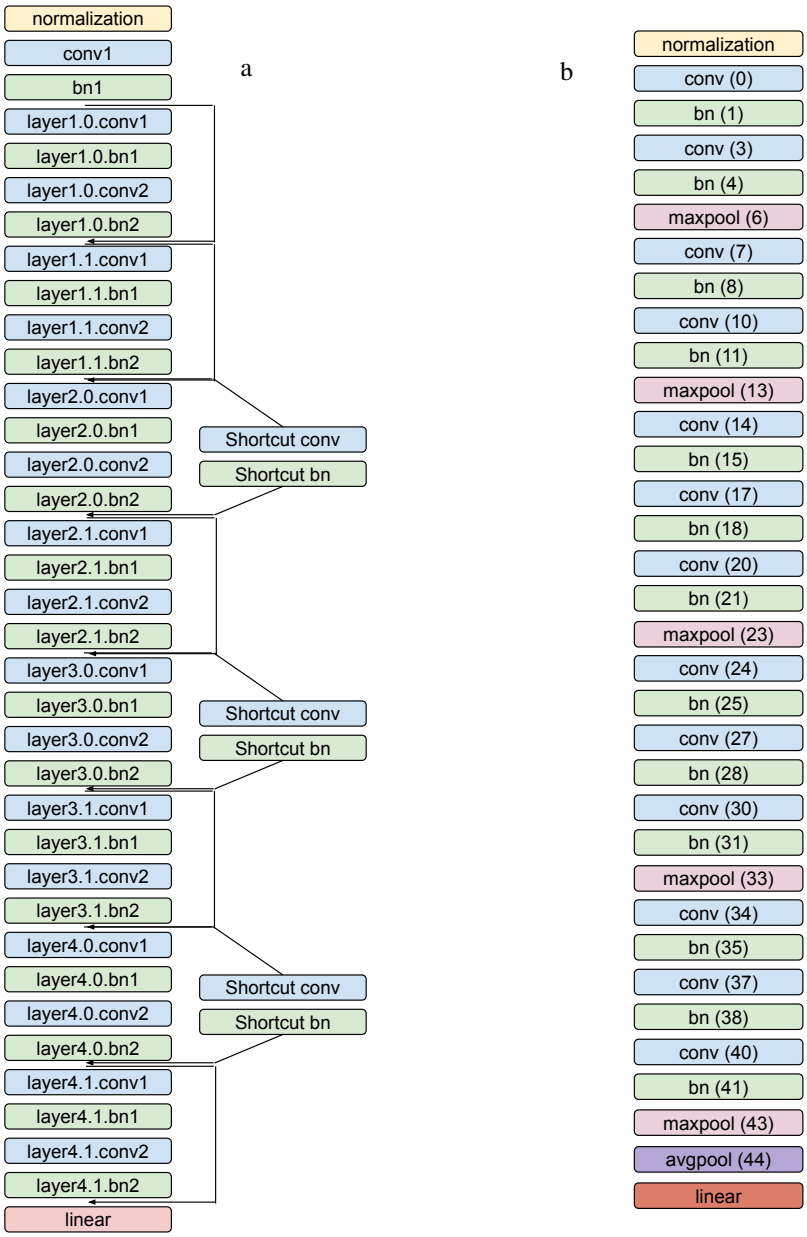

Figure 7: Architectures and layer names for **(a)** ResNet **(b)** VGG

A.3. Tables

In this section, we provide a series of tables showing how performance measures such as loss function value and accuracy (for clean and attacked images) vary with different variants of partial adversarial training.

An empty cell means network could not learn and defaults to random guessing (10% accuracy). We leave out results for such cases to avoid misrepresenting the trends in the data.

For the later/earlier layers retraining tables (Tables 1-8) first row (conv1 for ResNet and conv(0) for VGG) means cutoff is before the first layer and last row (all layers) means cutoff is after the last layer. For later layer retraining, cutoff before the first layer means nothing is frozen (every layer is retrained), cutoff after last layer means all layers are frozen (nothing is retrained). For earlier layer retraining, cutoff before the first layer means all layers are frozen (nothing is retrained), cutoff after last layer means nothing is frozen (every layer is retrained). Even though these correspond to all natural or all adversarial models, we include them for completeness i.e. sweeping the cutoff before and after every layer in the networks.

Main takeaways from the extended tables are:

- The freezing, reinitializing and retraining process affects the networks in terms of accuracy and loss, even if the retraining is done in the same type as the initial training. This shows the effect of retraining and the effect of partial training are separate, and justifies our decision to report the *differences* in accuracy and loss.
- In later layer retraining tables (Tables 1-4), for natural initial training and adversarial retraining (N1A2 columns), after some cutoff point the network is unable to learn and defaults to random guessing. As a result, we omit plots corresponding to cases that use these columns.
- In earlier layer retraining tables (Tables 5-8), the columns corresponding to empty columns in later retraining tables (N2A1) exhibit steep collapse of adversarial loss and accuracy. This is another testament to the importance of earlier layers in adversarial training. Conversely, columns A1N2 or A2N1 don't decline as sharply, when the same number of natural layers are located at the end of the network.
- In single layer retraining tables (Tables 9-12), retraining single convolutional layers affects losses and accuracies significantly more than retraining single batch norm layers.

Table 1: Later Layer Retraining Losses (ResNet)

	Clean Losses				Adversarial Losses			
cutoff before	**N1N2**	**N1A2**	**A1A2**	**A1N2**	**N1N2**	**N1A2**	**A1A2**	**A1N2**
conv1(nothing)	0.1945	0.5014	0.5014	0.1945	38.0540	1.6269	1.6269	38.0540
bn1	0.2049	0.4955	0.4993	0.2081	37.8757	1.6730	1.6819	37.9625
layer1.0.conv1	0.2053	0.4977	0.4985	0.2094	35.8906	1.6846	1.6613	36.1258
layer1.0.bn1	0.1966	0.5069	0.5022	0.2099	38.4588	1.6925	1.6835	36.3031
layer1.0.conv2	0.2026	0.5124	0.5009	0.2106	39.5354	1.6604	1.6971	37.9730
layer1.0.bn2	0.2151	0.5034	0.4992	0.2172	39.7665	1.6898	1.7274	30.1866
layer1.1.conv1	0.2067	0.5175	0.4957	0.2222	40.5081	1.7216	1.6955	29.2344
layer1.1.bn1	0.2236	0.5421	0.4972	0.2493	43.0724	1.7266	1.7041	27.0912
layer1.1.conv2	0.2280	0.5498	0.4911	0.2586	49.8329	1.7131	1.6594	27.0572
layer1.1.bn2	0.2299	0.5513	0.4906	0.2795	47.2077	1.6918	1.7189	20.1859
layer2.0.conv1	0.2242	0.5451	0.4958	0.2906	44.2159	1.7822	1.7392	19.7713
layer2.0.bn1	0.2367	0.5721	0.4916	0.3040	51.4650	1.8189	1.7212	17.5995
layer2.0.conv2	0.2304	0.5942	0.4874	0.3005	50.0523	1.8563	1.7349	17.1955
layer2.0.bn2	0.2620	0.6148	0.4874	0.3304	53.2214	1.8274	1.7526	16.4190
layer2.0.shortcut.0	0.2631	0.6194	0.4872	0.3109	46.8789	1.8617	1.7586	17.5351
layer2.0.shortcut.1	0.2671	0.6572	0.4914	0.3149	50.4867	1.9373	1.7639	14.8572
layer2.1.conv1	0.2504	0.8359	0.4910	0.3612	45.8021	2.1616	1.7648	13.2777
layer2.1.bn1	0.2727	0.9263	0.4873	0.3985	54.6465	2.2125	1.7820	13.4962
layer2.1.conv2	0.2682	0.9190	0.4927	0.4094	55.8137	2.2266	1.7532	12.0238
layer2.1.bn2	0.2846	0.9864	0.4857	0.4580	57.3148	2.1707	1.7394	12.1764
layer3.0.conv1	0.2619	1.0163	0.4789	0.4179	57.0524	2.3927	1.7731	10.7359
layer3.0.bn1	0.2862	1.1751	0.4910	0.4565	57.9531	2.4037	1.7746	9.8076
layer3.0.conv2	0.2850	1.5562	0.4823	0.4904	55.9327	2.3852	1.8091	9.9544
layer3.0.bn2	0.2846		0.4916	0.5162	58.0367		1.9031	9.5772
layer3.0.shortcut.0	0.2829		0.4961	0.5201	55.5235		1.8557	9.8108
layer3.0.shortcut.1	0.2796		0.4976	0.5227	55.8342		1.9033	7.8200
layer3.1.conv1	0.2831		0.4995	0.5391	57.6188		1.8174	7.6395
layer3.1.bn1	0.2758		0.4925	0.5479	50.1592		2.0214	6.8201
layer3.1.conv2	0.2997		0.4891	0.5985	49.0586		2.0156	7.1408
layer3.1.bn2	0.2723		0.4936	0.6047	36.4001		2.1413	6.6405
layer4.0.conv1	0.2719		0.4885	0.5909	36.9200		2.1736	6.8001
layer4.0.bn1	0.2600		0.4949	0.5983	38.4556		2.2224	6.2694
layer4.0.conv2	0.2619		0.4863	0.5978	37.9774		2.2110	6.2133
layer4.0.bn2	0.2432		0.4963	0.6234	39.4737		2.3133	5.9663
layer4.0.shortcut.0	0.2426		0.5022	0.6430	38.8861		2.3768	6.0537
layer4.0.shortcut.1	0.2377		0.4981	0.6583	39.4901		2.3287	5.9114
layer4.1.conv1	0.2374		0.5070	0.6722	39.9266		2.3761	5.9372
layer4.1.bn1	0.2256		0.4899	0.5998	37.4746		2.0082	4.3841
layer4.1.conv2	0.2250		0.4900	0.6431	38.1292		2.0626	4.6984
layer4.1.bn2	0.2144		0.4939	0.5417	40.2783		1.7630	3.1468
linear	0.2071		0.4918	0.4798	40.6797		1.7274	2.3212
all layers	0.1945	0.1945	0.5014	0.5014	38.0540	38.0540	1.6269	1.6269

Table 2: Later Layer Retraining Losses (VGG)

	Clean Losses				**Adversarial Losses**			
cutoff before	**N1N2**	**N1A2**	**A1A2**	**A1N2**	**N1N2**	**N1A2**	**A1A2**	**A1N2**
conv (0)	0.4980	0.6599	0.6599	0.498	57.8878	2.3842	2.3842	57.8878
bn (1)	0.5181	0.6714	0.6574	0.4990	61.1712	2.3564	2.3276	57.7251
conv (3)	0.4575	0.6687	0.6569	0.5253	63.6476	2.2664	2.4131	41.5692
bn (4)	0.5278	0.6765	0.6579	0.6287	63.0192	2.3538	2.3517	34.5760
conv (7)	0.5153	0.7292	0.6607	0.6626	69.8560	2.2335	2.3543	26.4202
bn (8)	0.5238	0.7586	0.6620	0.7769	66.2338	2.3331	2.4048	21.5849
conv (10)	0.5162	1.1720	0.6586	0.7888	66.5830	2.0850	2.3299	20.4872
bn (11)	0.5437		0.6527	0.9070	61.3882		2.4591	16.1664
conv (14)	0.5380		0.6637	0.9890	63.9610		2.4876	15.4178
bn (15)	0.5843		0.6787	1.1269	60.3823		2.8001	13.1493
conv (17)	0.5820		0.6736	1.1378	61.1271		2.7076	12.7263
bn (18)	0.5952		0.7232	1.2576	56.8554		3.3437	11.0207
conv (20)	0.6109		0.7092	1.2483	58.3716		3.2287	10.9078
bn (21)	0.6442		0.7796	1.3160	69.7752		3.8246	9.7352
conv (24)	0.6789		0.7585	1.3025	74.6683		3.7674	9.7130
bn (25)	0.7297		0.7639	1.3776	81.8649		3.7466	9.8278
conv (27)	0.7403		0.7712	1.3997	80.8411		3.7971	9.8307
bn (28)	0.8128		0.7223	1.3968	78.9955		3.1944	9.7534
conv (30)	0.8157		0.7193	1.3968	78.1335		3.1774	9.7724
bn (31)	0.8382		0.6936	1.0940	78.4322		2.7522	7.5895
conv (34)	0.8121		0.6916	1.0704	70.0127		2.7288	7.3802
bn (35)	0.8100		0.6920	0.9646	86.9406		2.6763	6.1586
conv (37)	0.8034		0.6898	0.9476	82.9781		2.6648	6.0080
bn (38)	0.8105		0.6791	0.8996	109.1270		2.5773	5.4172
conv (40)	0.8041		0.6951	0.8912	103.2717		2.7158	5.3210
bn (41)	0.8382		0.6983	0.8824	93.1972		2.7545	4.9857
linear	0.7677		0.6991	0.9259	88.7032		2.7726	5.4345
all layers	0.4980	0.4980	0.6599	0.6599	57.8878	57.8878	2.3842	2.3842

Table 3: Later Layer Retraining Accuracies (ResNet)

	Clean Losses				**Adversarial Losses**			
cutoff before	**N1N2**	**N1A2**	**A1A2**	**A1N2**	**N1N2**	**N1A2**	**A1A2**	**A1N2**
conv1(nothing)	94.53	84.01	84.01	94.53	0.00	48.38	48.38	0.00
bn1	94.33	84.36	84.26	93.85	0.00	47.59	48.36	0.00
layer1.0.conv1	94.12	83.97	84.07	93.75	0.00	47.11	47.88	0.00
layer1.0.bn1	94.39	83.59	83.89	94.10	0.00	46.68	48.03	0.00
layer1.0.conv2	94.29	83.89	84.29	93.91	0.00	46.76	48.47	0.00
layer1.0.bn2	94.18	83.95	83.84	93.59	0.00	46.86	48.01	0.03
layer1.1.conv1	94.25	83.68	84.38	93.62	0.00	45.31	48.17	0.09
layer1.1.bn1	93.86	82.82	84.10	92.91	0.00	44.72	48.21	0.25
layer1.1.conv2	93.80	83.03	84.27	92.78	0.00	44.97	48.07	0.21
layer1.1.bn2	93.99	82.76	84.32	92.11	0.00	44.57	47.86	0.91
layer2.0.conv1	93.94	83.02	84.12	92.03	0.00	43.10	47.53	1.43
layer2.0.bn1	93.93	82.25	84.34	91.80	0.00	42.08	47.62	2.31
layer2.0.conv2	93.98	81.68	84.37	91.73	0.00	40.09	47.81	2.14
layer2.0.bn2	93.46	80.85	84.23	91.31	0.00	39.74	47.52	3.28
layer2.0.shortcut.0	93.24	81.07	84.51	91.40	0.00	38.63	47.92	2.34
layer2.0.shortcut.1	93.36	80.41	84.24	91.34	0.00	35.57	47.10	4.62
layer2.1.conv1	93.76	77.15	84.66	90.43	0.00	24.50	47.40	6.86
layer2.1.bn1	93.55	74.82	84.24	89.79	0.00	21.79	47.08	8.48
layer2.1.conv2	93.50	75.07	84.45	89.57	0.00	21.52	47.24	10.23
layer2.1.bn2	93.59	72.89	84.77	88.93	0.00	21.98	47.68	12.27
layer3.0.conv1	93.90	72.48	85.12	89.51	0.00	17.45	47.25	12.47
layer3.0.bn1	93.45	68.35	84.57	88.71	0.00	14.89	47.39	15.81
layer3.0.conv2	93.44	57.44	84.67	87.96	0.00	10.84	46.90	16.44
layer3.0.bn2	93.96		84.84	87.56	0.00		46.43	19.13
layer3.0.shortcut.0	93.67		84.49	87.78	0.00		46.84	18.33
layer3.0.shortcut.1	93.73		84.38	87.25	0.00		46.68	22.97
layer3.1.conv1	93.61		84.63	86.93	0.00		46.98	23.85
layer3.1.bn1	94.03		84.51	85.95	0.00		46.02	26.73
layer3.1.conv2	93.67		84.74	85.65	0.00		45.83	26.45
layer3.1.bn2	94.09		84.78	85.99	0.00		45.73	29.64
layer4.0.conv1	94.02		84.67	86.18	0.00		45.13	29.34
layer4.0.bn1	94.26		84.89	85.96	0.00		45.54	32.13
layer4.0.conv2	94.30		84.93	86.01	0.00		45.62	31.86
layer4.0.bn2	94.45		84.72	85.60	0.00		45.57	34.06
layer4.0.shortcut.0	94.46		84.47	85.28	0.00		45.05	33.86
layer4.0.shortcut.1	94.50		84.77	84.90	0.00		45.08	34.29
layer4.1.conv1	94.43		84.52	85.21	0.00		44.79	34.49
layer4.1.bn1	94.45		84.73	85.04	0.00		46.49	38.11
layer4.1.conv2	94.57		84.78	84.50	0.00		46.08	37.58
layer4.1.bn2	94.43		84.43	84.90	0.00		47.75	42.15
linear	94.41		84.42	84.90	0.00		48.06	45.05
all layers	94.53	94.53	84.01	84.01	0.00	0.00	48.38	48.38

Table 4: Later Layer Retraining Accuracies (VGG)

	Clean Losses				Adversarial Losses			
cutoff before	**N1N2**	**N1A2**	**A1A2**	**A1N2**	**N1N2**	**N1A2**	**A1A2**	**A1N2**
conv (0)	92.49	79.91	79.91	92.49	0.00	42.46	42.46	0.00
bn (1)	92.12	79.53	79.75	92.46	0.00	41.93	42.44	0.00
conv (3)	92.83	79.34	80.01	91.92	0.00	41.89	42.65	0.00
bn (4)	92.35	79.38	79.91	90.53	0.00	36.60	42.39	0.27
conv (7)	91.97	78.16	79.82	89.99	0.00	33.45	42.77	1.77
bn (8)	92.47	77.43	80.25	88.33	0.00	30.35	42.22	3.60
conv (10)	92.20	63.96	79.52	88.08	0.00	22.65	42.60	5.50
bn (11)	92.18		80.55	86.35	0.00		42.22	11.26
conv (14)	92.17		80.09	85.87	0.00		42.13	12.65
bn (15)	91.78		80.19	84.35	0.00		40.96	18.93
conv (17)	91.81		80.13	83.61	0.00		41.68	19.27
bn (18)	92.11		80.19	82.30	0.00		39.77	24.35
conv (20)	92.18		80.15	82.19	0.00		40.55	24.88
bn (21)	92.24		80.08	81.59	0.00		39.50	27.80
conv (24)	92.32		80.12	81.61	0.00		39.68	27.11
bn (25)	92.45		80.07	81.14	0.00		39.64	29.12
conv (27)	92.51		80.22	80.83	0.01		39.59	29.34
bn (28)	92.62		80.17	80.47	0.00		41.54	29.89
conv (30)	92.56		80.11	80.79	0.00		41.35	30.00
bn (31)	92.55		79.85	80.49	0.00		42.62	32.00
conv (34)	92.53		79.82	80.51	0.01		42.43	32.38
bn (35)	92.57		79.86	80.89	0.54		42.56	35.10
conv (37)	92.50		79.84	80.84	0.01		42.59	35.30
bn (38)	92.52		79.94	80.86	0.72		42.51	36.75
conv (40)	92.55		79.88	80.85	0.46		42.40	36.92
bn (41)	92.53		79.86	81.18	0.00		42.70	38.54
linear	92.48		79.84	81.15	0.00		42.71	38.51
all layers	92.49	92.49	79.91	79.91	0.00	0.00	42.46	42.46

Table 5: Earlier Layer Retraining Losses (ResNet)

	Clean Losses				Adversarial Losses			
cutoff before	**N2N1**	**N2A1**	**A2A1**	**A2N1**	**N2N1**	**N2A1**	**A2A1**	**A2N1**
conv1(nothing)	0.1945	0.5014	0.5014	0.1945	38.0540	1.6269	1.6269	38.0540
bn1	0.1936	0.4653	0.5103	2.0804	37.7234	3.9362	1.6993	14.5946
layer1.0.conv1	0.2219	0.4389	0.4995	2.2553	40.5747	4.2964	1.6520	2.3752
layer1.0.bn1	0.1996	0.4041	0.4987	10.4538	37.9801	9.1672	1.6722	11.5121
layer1.0.conv2	0.2003	0.4007	0.5005	2.4423	38.0250	9.2057	1.6854	2.6324
layer1.0.bn2	0.1994	0.3777	0.4983	2.4522	37.4451	13.6233	1.6382	2.5428
layer1.1.conv1	0.1977	0.3738	0.4987	2.0894	37.7095	13.1055	1.6425	2.1809
layer1.1.bn1	0.1994	0.3565	0.4993	1.8049	37.6339	16.7354	1.6474	2.0890
layer1.1.conv2	0.2052	0.3577	0.4991	2.5567	38.0990	16.9609	1.6450	3.0294
layer1.1.bn2	0.2015	0.3460	0.4984	2.6859	37.5756	19.7851	1.6455	3.3587
layer2.0.conv1	0.2009	0.3440	0.4987	2.1813	37.8375	18.2050	1.6430	2.2330
layer2.0.bn1	0.2058	0.3239	0.4987	1.7039	37.5928	21.4722	1.6438	2.0087
layer2.0.conv2	0.2064	0.3281	0.4997	2.2398	38.0195	21.0103	1.6453	2.6578
layer2.0.bn2	0.2040	0.2962	0.4981	1.6736	37.8628	25.4394	1.6478	2.0592
layer2.0.shortcut.0	0.1997	0.3021	0.4976	1.7098	38.2652	25.9670	1.6461	2.0919
layer2.0.shortcut.1	0.2033	0.2984	0.4976	1.1960	38.0837	26.0197	1.6418	1.7599
layer2.1.conv1	0.2004	0.2985	0.4986	1.4597	37.6149	26.6355	1.6480	1.9442
layer2.1.bn1	0.2044	0.2796	0.4987	1.6041	38.0842	28.6019	1.6526	2.0155
layer2.1.conv2	0.2095	0.2743	0.4975	1.2086	37.8631	28.8347	1.6485	1.7346
layer2.1.bn2	0.2112	0.2567	0.5002	1.3181	37.7653	31.1661	1.6552	1.7783
layer3.0.conv1	0.2079	0.2635	0.5000	1.4755	37.6879	31.0587	1.6551	1.9180
layer3.0.bn1	0.2088	0.2408	0.5011	0.8774	37.5209	31.4367	1.6639	1.5662
layer3.0.conv2	0.2099	0.2402	0.5029	0.9044	37.7449	29.9869	1.6624	1.6063
layer3.0.bn2	0.2144	0.2226	0.5006	0.7738	37.2279	31.1123	1.6712	1.5213
layer3.0.shortcut.0	0.2147	0.2205	0.4999	0.7039	37.6779	30.9538	1.6715	1.4656
layer3.0.shortcut.1	0.2050	0.2149	0.4983	0.7257	37.3455	31.2024	1.6656	1.4573
layer3.1.conv1	0.2136	0.2114	0.4992	0.7043	37.6654	31.2087	1.6610	1.4849
layer3.1.bn1	0.2082	0.2094	0.5017	0.7049	36.2429	31.8986	1.6617	1.4577
layer3.1.conv2	0.2071	0.1969	0.5016	0.6895	36.7227	31.1560	1.6632	1.4475
layer3.1.bn2	0.2007	0.2041	0.5050	0.6825	35.5956	32.0541	1.6713	1.4619
layer4.0.conv1	0.1995	0.1936	0.5062	0.6939	36.2526	29.9096	1.6667	1.4380
layer4.0.bn1	0.1943	0.1998	0.5143	0.5951	34.5393	31.3105	1.6633	1.4253
layer4.0.conv2	0.1982	0.1994	0.5119	0.5848	33.9057	30.2159	1.6521	1.4290
layer4.0.bn2	0.1877	0.1904	0.5153	0.5330	32.7324	35.7960	1.6445	1.4598
layer4.0.shortcut.0	0.1840	0.1884	0.5107	0.5519	32.1108	36.0153	1.6207	1.4338
layer4.0.shortcut.1	0.1900	0.2011	0.5069	0.5393	33.2710	35.6191	1.6210	1.4561
layer4.1.conv1	0.1851	0.1844	0.5130	0.5308	34.4472	36.4343	1.6141	1.4473
layer4.1.bn1	0.2135	0.2051	0.5248	0.5192	42.0783	40.0391	1.5661	1.5053
layer4.1.conv2	0.1960	0.1970	0.5273	0.5175	35.5188	35.8699	1.5364	1.4871
layer4.1.bn2	0.2266	0.1994	0.5326	0.5202	52.8249	38.6505	1.5361	1.5318
linear	0.2047	0.2123	0.5247	0.5204	48.3074	48.7007	1.5587	1.5157
all layers	0.1945	0.1945	0.5014	0.5014	38.0540	38.0540	1.6269	1.6269

Table 6: Earlier Layer Retraining Losses (VGG)

	Clean Losses				Adversarial Losses			
cutoff before	**N2N1**	**N2A1**	**A2A1**	**A2N1**	**N2N1**	**N2A1**	**A2A1**	**A2N1**
conv (0)	0.4980	0.6599	0.6599	0.4980	57.8878	2.3842	2.3842	57.8878
bn (1)	0.5107	0.5922	0.6626	5.9573	56.9417	5.8432	2.4233	23.4814
conv (3)	0.5119	0.5744	0.6626	7.2372	56.9679	6.3913	2.4395	19.6369
bn (4)	0.5147	0.5228	0.6625	3.9839	56.8867	19.5412	2.3935	5.7267
conv (7)	0.5244	0.5220	0.6621	3.5687	57.2322	23.9488	2.3954	5.0155
bn (8)	0.5349	0.4804	0.6653	2.8108	56.4171	33.8566	2.4084	2.8111
conv (10)	0.5297	0.4804	0.6639	4.0320	55.9973	33.7482	2.4009	5.5605
bn (11)	0.5320	0.4432	0.6666	3.0262	55.0885	40.1923	2.4195	4.9806
conv (14)	0.5381	0.4273	0.6676	1.9984	54.5558	42.8679	2.4161	3.3679
bn (15)	0.5005	0.4101	1.0006	1.2732	54.9113	45.7596	1.9656	2.3153
conv (17)	0.4965	0.3997	0.7193	1.8419	55.8193	45.2905	2.3945	3.9014
bn (18)	0.4559	0.3512	0.7042	1.4556	56.5107	46.7417	2.1502	3.3016
conv (20)	0.4292	0.3400	0.6969	1.4435	58.8676	46.8702	2.1686	3.3421
bn (21)	0.4220	0.3186	0.6977	0.8188	57.1714	44.4637	1.9327	1.8160
conv (24)	0.4109	0.3176	0.6930	0.8356	52.6446	44.2168	1.9985	1.9317
bn (25)	0.3717	0.3181	0.6730	0.9687	53.6615	41.6764	1.9269	2.6687
conv (27)	0.3848	0.3455	0.6750	0.8107	54.9071	39.9270	1.9731	1.9800
bn (28)	0.3826	0.3587	0.6492	0.7263	55.5252	31.9606	2.1764	1.6357
conv (30)	0.3904	0.3498	0.6458	0.7080	54.9429	29.7600	2.1506	1.6369
bn (31)	0.4127	0.3712	0.6532	0.6319	53.2463	25.5389	2.2302	1.8272
conv (34)	0.4149	0.3717	0.6560	0.7130	52.4027	26.3126	2.2190	1.7283
bn (35)	0.4336	0.3704	0.6481	0.6273	54.7199	23.1462	2.3984	2.1159
conv (37)	0.4123	0.3744	0.6426	0.6486	52.1497	22.6216	2.2707	2.0604
bn (38)	0.4070	0.3919	0.6531	0.6394	44.0254	25.1886	2.4161	2.3111
conv (40)	0.4164	0.3824	0.6502	0.6249	42.7574	25.0965	2.3500	2.3019
bn (41)	0.4585	0.4154	0.6235	0.6317	44.7905	34.0194	2.3297	2.3805
linear	0.4992	0.4782	0.6607	0.6417	53.4132	48.9287	2.4837	2.5547
all layers	0.4980	0.4980	0.6599	0.6599	57.8878	57.8878	2.3842	2.3842

Table 7: Earlier Layer Retraining Accuracies (ResNet)

	Clean Losses				Adversarial Losses			
cutoff before	**N2N1**	**N2A1**	**A2A1**	**A2N1**	**N2N1**	**N2A1**	**A2A1**	**A2N1**
conv1(nothing)	94.53	84.01	84.01	94.53	0.00	48.38	48.38	0.00
bn1	94.53	85.62	83.53	48.80	0.00	13.44	46.81	0.51
layer1.0.conv1	93.78	86.21	83.93	15.04	0.00	13.87	47.83	9.33
layer1.0.bn1	94.34	87.28	84.07	11.31	0.00	1.95	47.46	6.00
layer1.0.conv2	94.24	87.32	83.92	11.34	0.00	2.04	47.07	8.03
layer1.0.bn2	94.36	88.36	84.03	9.91	0.00	0.39	48.09	5.75
layer1.1.conv1	94.34	88.42	83.92	23.84	0.00	0.68	47.77	19.93
layer1.1.bn1	94.54	88.71	83.91	34.31	0.00	0.15	47.73	21.76
layer1.1.conv2	94.16	88.80	83.97	10.45	0.00	0.20	47.79	9.77
layer1.1.bn2	94.51	89.34	84.03	11.38	0.00	0.08	47.86	7.94
layer2.0.conv1	94.36	89.39	84.04	16.98	0.00	0.21	47.92	13.58
layer2.0.bn1	94.21	89.78	83.94	37.85	0.00	0.10	47.97	23.65
layer2.0.conv2	94.17	89.72	83.98	19.29	0.00	0.12	47.86	8.31
layer2.0.bn2	94.29	90.65	83.93	40.90	0.00	0.05	48.01	22.07
layer2.0.shortcut.0	94.38	90.60	83.94	40.27	0.00	0.02	48.06	20.95
layer2.0.shortcut.1	94.26	90.85	83.82	60.27	0.00	0.04	48.19	32.96
layer2.1.conv1	94.35	90.57	83.95	51.40	0.00	0.02	48.00	26.72
layer2.1.bn1	94.11	91.49	83.84	43.94	0.00	0.01	48.03	24.24
layer2.1.conv2	94.16	91.46	83.86	59.94	0.00	0.01	48.19	33.65
layer2.1.bn2	93.98	91.95	83.81	55.16	0.00	0.00	48.06	31.94
layer3.0.conv1	94.07	91.72	83.77	51.95	0.00	0.00	47.79	29.83
layer3.0.bn1	93.81	92.48	83.86	71.80	0.00	0.00	47.82	41.06
layer3.0.conv2	94.05	92.23	83.76	70.94	0.00	0.00	47.67	39.32
layer3.0.bn2	93.70	92.89	83.81	75.61	0.00	0.00	47.25	43.17
layer3.0.shortcut.0	93.71	92.95	83.98	78.11	0.00	0.00	47.26	44.97
layer3.0.shortcut.1	94.05	93.17	83.89	77.66	0.00	0.00	47.49	45.35
layer3.1.conv1	93.73	93.52	83.80	78.12	0.00	0.02	47.54	44.08
layer3.1.bn1	93.65	93.14	83.83	78.33	0.00	0.02	47.32	45.24
layer3.1.conv2	93.95	93.78	83.80	78.30	0.00	0.01	47.34	45.98
layer3.1.bn2	93.85	93.52	83.80	78.88	0.00	0.00	47.29	45.61
layer4.0.conv1	94.03	93.75	83.82	78.61	0.00	0.05	47.44	45.81
layer4.0.bn1	93.90	93.93	83.45	81.55	0.00	0.02	47.08	47.55
layer4.0.conv2	93.76	93.92	83.71	81.49	0.00	0.02	47.33	47.24
layer4.0.bn2	94.24	93.99	83.34	83.42	0.00	0.00	47.07	48.25
layer4.0.shortcut.0	94.26	94.07	83.57	82.98	0.00	0.01	47.45	48.38
layer4.0.shortcut.1	94.07	93.99	83.58	83.22	0.00	0.00	47.73	48.57
layer4.1.conv1	94.30	94.12	83.61	83.22	0.00	0.00	47.59	48.59
layer4.1.bn1	94.09	94.11	83.45	83.42	0.00	0.00	48.72	48.86
layer4.1.conv2	94.30	94.41	83.69	83.96	0.00	0.00	48.87	48.86
layer4.1.bn2	94.07	94.13	83.06	83.45	0.00	0.00	49.07	48.76
linear	94.11	94.04	83.48	83.27	0.00	0.00	49.02	49.08
all layers	94.53	94.53	84.01	84.01	0.00	0.00	48.38	48.38

Table 8: Earlier Layer Retraining Accuracies (VGG)

	Clean Losses				Adversarial Losses			
cutoff before	**N2N1**	**N2A1**	**A2A1**	**A2N1**	**N2N1**	**N2A1**	**A2A1**	**A2N1**
conv (0)	92.49	79.91	79.91	92.49	0.00	42.46	42.46	0.00
bn (1)	92.27	81.71	79.90	34.98	0.00	11.95	41.98	0.67
conv (3)	92.27	82.22	79.85	26.64	0.00	10.54	41.83	1.46
bn (4)	92.13	83.78	79.91	10.29	0.00	0.28	42.52	9.95
conv (7)	92.24	83.85	79.90	10.27	0.00	0.01	42.34	9.95
bn (8)	91.99	85.30	79.81	10.00	0.00	0.01	42.17	10.00
conv (10)	92.03	85.46	79.83	13.83	0.00	0.00	42.21	10.74
bn (11)	91.58	87.00	79.70	27.17	0.00	0.00	42.53	14.89
conv (14)	91.67	87.34	79.80	37.22	0.00	0.00	42.15	16.10
bn (15)	91.40	89.08	67.35	62.42	0.00	0.00	38.32	30.63
conv (17)	91.66	89.20	77.69	58.72	0.00	0.00	39.95	27.74
bn (18)	91.39	91.33	77.84	65.53	0.00	0.00	41.79	31.38
conv (20)	91.75	91.33	78.25	68.14	0.00	0.01	41.65	33.75
bn (21)	92.18	92.53	77.95	73.79	0.00	0.01	42.90	38.68
conv (24)	92.61	92.45	78.39	74.91	0.00	0.00	42.62	39.34
bn (25)	92.74	93.11	78.16	76.43	0.00	0.00	43.34	40.40
conv (27)	92.71	92.77	78.63	76.49	0.00	0.00	42.97	40.44
bn (28)	92.83	92.72	79.78	77.47	0.00	0.07	41.82	41.86
conv (30)	92.68	93.20	79.98	77.95	0.00	0.00	41.92	41.85
bn (31)	92.88	92.89	79.75	79.27	0.00	0.40	41.62	40.98
conv (34)	92.70	92.84	79.57	77.72	0.00	0.58	42.00	41.67
bn (35)	92.56	92.75	80.34	80.56	0.00	2.02	41.35	41.38
conv (37)	92.89	92.94	79.91	79.52	0.00	0.85	42.38	41.12
bn (38)	92.98	92.90	80.54	79.98	0.00	0.00	41.55	41.51
conv (40)	92.78	93.19	80.29	80.42	0.00	0.00	41.78	41.77
bn (41)	92.59	92.98	80.62	80.65	0.00	0.00	40.90	41.20
linear	92.32	92.93	80.30	80.61	0.00	0.00	42.06	40.53
all layers	92.49	92.49	79.91	79.91	0.00	0.00	42.46	42.46

Table 9: Single Layer Retraining Losses (ResNet)
N_s:single layer natural, N_r:rest of the network natural
A_s:single layer adversarial, A_r:rest of the network adversarial

	Clean Losses				Adversarial Losses			
retrained layer	$\mathbf{N_sN_r}$	$\mathbf{N_sA_r}$	$\mathbf{A_sA_r}$	$\mathbf{A_sN_r}$	$\mathbf{N_sN_r}$	$\mathbf{N_sA_r}$	$\mathbf{A_sA_r}$	$\mathbf{A_sN_r}$
conv1	0.1940	0.4633	0.4994	1.6169	37.7088	3.9250	1.6336	14.7941
bn1	0.1946	0.4761	0.4993	1.3837	38.0591	2.2088	1.6337	11.6455
layer1.0.conv1	0.1948	0.4329	0.4991	1.4610	37.8514	4.0192	1.6391	10.4059
layer1.0.bn1	0.1947	0.4916	0.4989	0.6380	37.9984	2.0390	1.6378	16.6529
layer1.0.conv2	0.1919	0.4236	0.4985	2.3810	37.6244	4.5307	1.6346	3.1375
layer1.0.bn2	0.1956	0.4745	0.4997	2.2241	37.6642	2.1243	1.6356	12.2359
layer1.1.conv1	0.1960	0.4365	0.4987	0.7633	37.7234	3.4764	1.6379	14.1267
layer1.1.bn1	0.1950	0.5021	0.4999	0.4337	38.0830	1.6949	1.6342	18.2846
layer1.1.conv2	0.1945	0.4464	0.4981	0.6821	37.6097	3.0051	1.6351	15.8009
layer1.1.bn2	0.1950	0.4876	0.4999	0.5573	37.8179	1.7700	1.6346	17.1894
layer2.0.conv1	0.1951	0.3864	0.4972	2.4113	37.9497	4.8081	1.6374	3.5114
layer2.0.bn1	0.1946	0.4707	0.4990	0.2995	38.2237	1.8608	1.6369	21.0361
layer2.0.conv2	0.1960	0.3674	0.4954	2.3035	38.4281	4.4937	1.6400	3.2908
layer2.0.bn2	0.1963	0.4697	0.4998	0.8513	38.3560	1.836	1.6348	20.4987
layer2.0.shortcut.0	0.1940	0.4541	0.4973	0.3473	38.2403	2.3443	1.6347	19.9055
layer2.0.shortcut.1	0.1941	0.4665	0.4988	0.7192	38.2570	1.8593	1.6333	19.8157
layer2.1.conv1	0.1947	0.4059	0.4949	2.2106	38.4508	3.2461	1.6416	14.1217
layer2.1.bn1	0.1966	0.5136	0.4988	0.3343	38.2437	1.5988	1.6349	22.1184
layer2.1.conv2	0.1958	0.4147	0.4934	2.2279	38.4952	2.8063	1.6416	14.1250
layer2.1.bn2	0.1950	0.4805	0.4978	2.4695	38.0967	1.7507	1.6385	9.5302
layer3.0.conv1	0.1910	0.3763	0.4935	2.2672	38.5363	3.8071	1.6451	7.0057
layer3.0.bn1	0.1947	0.4726	0.4971	2.3548	38.1648	1.7239	1.6373	4.4397
layer3.0.conv2	0.1975	0.3818	0.4888	2.1484	39.3820	4.1993	1.6681	5.9880
layer3.0.bn2	0.1980	0.4785	0.4962	2.7564	38.7571	1.7337	1.6457	7.7941
layer3.0.shortcut.0	0.1916	0.4379	0.4933	0.4799	38.5699	1.9685	1.6436	22.1319
layer3.0.shortcut.1	0.1948	0.4672	0.4957	2.6051	37.9700	1.7950	1.6428	11.8103
layer3.1.conv1	0.1971	0.4170	0.4870	1.8742	38.7524	3.2120	1.6645	12.1069
layer3.1.bn1	0.1955	0.5172	0.4978	0.5756	37.8325	1.5687	1.6383	21.0010
layer3.1.conv2	0.2023	0.4293	0.4858	2.0325	39.4832	2.8067	1.6909	8.2243
layer3.1.bn2	0.1960	0.4871	0.4954	2.6650	37.4280	1.7825	1.6470	10.0324
layer4.0.conv1	0.2080	0.4261	0.4847	1.8363	38.0785	3.5187	1.7034	4.3053
layer4.0.bn1	0.1978	0.4971	0.4962	1.6089	37.2622	1.6252	1.6431	12.5635
layer4.0.conv2	0.2256	0.4439	0.4757	2.0222	37.4409	3.7431	1.8479	4.4633
layer4.0.bn2	0.1982	0.4474	0.4934	2.4358	35.4980	2.1569	1.6589	6.8881
layer4.0.shortcut.0	0.1999	0.4498	0.4877	0.5912	38.0866	2.0395	1.6906	19.3393
layer4.0.shortcut.1	0.1995	0.4490	0.4935	2.3650	37.0167	2.1194	1.6585	7.4036
layer4.1.conv1	0.2342	0.4870	0.4758	2.1308	37.7290	3.7817	2.0426	2.6247
layer4.1.bn1	0.1986	0.4572	0.4898	2.0068	34.8453	1.7671	1.6768	3.9613
layer4.1.conv2	0.2229	0.4982	0.4806	2.0780	35.9666	3.0972	1.8754	2.8999
layer4.1.bn2	0.2115	0.5184	0.4963	2.2893	40.6968	2.7988	1.8779	2.7234
linear	0.2071	0.4798	0.4924	2.2959	40.6797	2.3212	1.8425	2.3437

Table 10: Single Layer Retraining Losses (VGG)
N_s:single layer natural, N_r:rest of the network natural
A_s:single layer adversarial, A_r:rest of the network adversarial

	Clean Losses				Adversarial Losses			
retrained layer	$\mathbf{N_sN_r}$	$\mathbf{N_sA_r}$	$\mathbf{A_sA_r}$	$\mathbf{A_sN_r}$	$\mathbf{N_sN_r}$	$\mathbf{N_sA_r}$	$\mathbf{A_sA_r}$	$\mathbf{A_sN_r}$
conv (0)	0.5099	0.5913	0.6620	5.6060	57.0806	5.9623	2.4143	23.1821
bn (1)	0.5019	0.6069	0.6599	1.9980	58.0907	3.1012	2.3854	25.0438
conv (3)	0.5131	0.5367	0.6607	3.7212	57.0017	13.0263	2.3860	5.1825
bn (4)	0.4996	0.6178	0.6601	1.4533	57.9325	2.6919	2.3848	27.6754
conv (7)	0.5288	0.5375	0.6613	2.9443	56.6551	9.0522	2.3919	4.7922
bn (8)	0.4985	0.6204	0.6598	1.7839	57.9536	2.5302	2.3856	27.5135
conv (10)	0.5162	0.5415	0.6618	2.7795	56.6342	8.2110	2.4037	6.6277
bn (11)	0.4958	0.6467	0.6597	3.4255	57.8814	2.4823	2.3870	3.4333
conv (14)	0.5219	0.6196	0.6609	2.7323	58.7280	8.5674	2.4217	9.8543
bn (15)	0.5014	0.6562	0.6591	3.4119	57.9910	2.5605	2.3890	3.4131
conv (17)	0.4992	0.7430	0.6620	2.5512	58.3959	8.7195	2.4610	16.2406
bn (18)	0.4996	0.6607	0.6591	3.5780	57.7582	2.8617	2.3884	3.6301
conv (20)	0.5064	0.8361	0.6614	2.0846	59.2473	8.2144	2.5322	12.4486
bn (21)	0.4972	0.6682	0.6590	3.5005	57.7088	3.1819	2.3929	3.5814
conv (24)	0.4977	0.9339	0.6647	2.4282	59.8478	7.9208	2.7008	11.0532
bn (25)	0.4950	0.6675	0.6595	3.4040	57.2956	3.4131	2.4026	3.7616
conv (27)	0.5060	0.9520	0.6784	2.6093	58.2348	7.2650	2.9325	6.4728
bn (28)	0.4993	0.6780	0.6602	3.3896	57.1871	3.4739	2.4206	3.3947
conv (30)	0.5163	0.9216	0.6748	2.4885	56.2977	6.4789	2.6982	5.8562
bn (31)	0.5018	0.6643	0.6611	1.5210	57.3601	3.3311	2.4359	9.9010
conv (34)	0.5390	0.8013	0.6669	2.3050	52.4877	5.0744	2.5002	3.8530
bn (35)	0.5049	0.6676	0.6620	1.9158	57.0209	3.2782	2.4393	4.4205
conv (37)	0.5484	0.7767	0.6686	2.2313	50.7040	4.5463	2.4905	2.3912
bn (38)	0.5204	0.6829	0.6618	2.6449	56.4117	3.3534	2.4024	3.3787
conv (40)	0.5609	0.7794	0.6743	2.2948	53.8834	4.3578	2.5333	2.3095
bn (41)	0.5980	0.8490	0.6756	2.3017	68.2466	4.5501	2.5697	2.3080
linear	0.6600	0.9025	0.6992	2.2837	81.1101	5.2264	2.7734	2.3184

Table 11: Single Layer Retraining Accuracies (ResNet)
N_s:single layer natural, N_r:rest of the network natural
A_s:single layer adversarial, A_r:rest of the network adversarial

	Clean Losses				**Adversarial Losses**			
retrained layer	$\mathbf{N_sN_r}$	$\mathbf{N_sA_r}$	$\mathbf{A_sA_r}$	$\mathbf{A_sN_r}$	$\mathbf{N_sN_r}$	$\mathbf{N_sA_r}$	$\mathbf{A_sA_r}$	$\mathbf{A_sN_r}$
conv1	94.49	85.58	84.01	58.34	0.00	14.09	48.14	0.44
bn1	94.45	85.41	83.98	66.87	0.00	34.92	48.19	3.05
layer1.0.conv1	94.34	86.33	83.92	67.00	0.00	15.99	48.02	6.73
layer1.0.bn1	94.47	84.55	84.01	80.45	0.00	37.63	47.98	0.11
layer1.0.conv2	94.40	86.86	84.02	16.93	0.00	13.40	48.18	9.71
layer1.0.bn2	94.43	85.18	84.00	44.22	0.00	36.26	48.07	0.28
layer1.1.conv1	94.51	86.37	83.97	79.56	0.00	21.35	48.07	1.65
layer1.1.bn1	94.49	84.43	83.93	85.73	0.00	44.27	48.19	0.01
layer1.1.conv2	94.37	85.83	84.00	80.74	0.00	24.73	48.07	0.41
layer1.1.bn2	94.49	84.81	84.03	82.30	0.00	42.55	48.17	0.06
layer2.0.conv1	94.44	87.98	84.09	10.01	0.00	15.79	48.06	0.01
layer2.0.bn1	94.48	85.24	84.03	89.72	0.00	41.33	47.95	0.00
layer2.0.conv2	94.40	88.01	84.13	12.99	0.00	18.52	48.13	0.00
layer2.0.bn2	94.49	85.06	83.97	71.69	0.00	42.00	48.18	0.00
layer2.0.shortcut.0	94.65	85.86	84.04	88.49	0.00	33.62	48.11	0.01
layer2.0.shortcut.1	94.50	85.41	84.02	75.65	0.00	41.48	48.15	0.00
layer2.1.conv1	94.60	87.04	84.08	19.81	0.00	26.94	48.11	0.00
layer2.1.bn1	94.40	83.95	84.01	88.42	0.00	46.21	48.15	0.00
layer2.1.conv2	94.50	86.49	84.27	19.37	0.00	31.28	48.20	0.00
layer2.1.bn2	94.47	84.81	84.11	10.21	0.00	43.91	48.07	0.01
layer3.0.conv1	94.51	88.01	84.14	11.04	0.00	24.76	48.11	0.00
layer3.0.bn1	94.57	84.97	84.23	10.28	0.00	44.07	48.21	0.03
layer3.0.conv2	94.38	87.38	84.27	26.56	0.00	23.83	47.95	0.00
layer3.0.bn2	94.44	85.14	84.19	10.09	0.00	44.34	48.07	4.20
layer3.0.shortcut.0	94.62	85.86	84.24	85.81	0.00	41.15	48.15	0.00
layer3.0.shortcut.1	94.48	85.10	84.13	16.53	0.00	43.59	48.01	0.00
layer3.1.conv1	94.20	86.39	84.05	38.68	0.00	30.05	47.46	0.00
layer3.1.bn1	94.52	83.69	84.22	80.01	0.00	47.40	48.13	0.00
layer3.1.conv2	94.49	86.08	84.41	34.02	0.00	34.33	47.74	0.00
layer3.1.bn2	94.39	84.36	84.04	10.14	0.00	44.23	48.08	0.51
layer4.0.conv1	94.50	86.29	84.28	54.69	0.00	32.56	47.62	0.00
layer4.0.bn1	94.28	84.21	84.21	40.36	0.00	46.09	48.14	0.00
layer4.0.conv2	94.40	86.40	84.70	46.84	0.00	34.71	46.11	0.00
layer4.0.bn2	94.43	85.20	84.13	14.41	0.00	43.17	47.87	0.00
layer4.0.shortcut.0	94.65	85.33	84.35	86.43	0.00	42.40	47.84	0.00
layer4.0.shortcut.1	94.46	84.69	84.13	19.39	0.00	42.91	47.81	0.00
layer4.1.conv1	94.53	85.49	84.37	35.18	0.00	37.14	45.10	0.20
layer4.1.bn1	94.45	84.73	84.41	26.71	0.00	44.87	47.77	0.00
layer4.1.conv2	94.61	84.88	84.59	75.54	0.00	39.17	46.75	0.00
layer4.1.bn2	94.46	84.81	84.22	11.10	0.00	44.18	47.38	1.07
linear	94.41	84.90	84.33	11.70	0.00	45.05	47.43	0.24

Table 12: Single Layer Retraining Accuracies (VGG)
N_s:single layer natural, N_r:rest of the network natural
A_s:single layer adversarial, A_r:rest of the network adversarial

	Clean Losses				Adversarial Losses			
retrained layer	$\mathbf{N_sN_r}$	$\mathbf{N_sA_r}$	$\mathbf{A_sA_r}$	$\mathbf{A_sN_r}$	$\mathbf{N_sN_r}$	$\mathbf{N_sA_r}$	$\mathbf{A_sA_r}$	$\mathbf{A_sN_r}$
conv (0)	92.35	81.98	79.82	36.48	0.00	11.62	42.24	0.90
bn (1)	92.56	81.71	79.94	71.56	0.00	30.00	42.44	0.85
conv (3)	92.47	83.47	79.90	10.68	0.00	2.14	42.43	9.70
bn (4)	92.58	81.29	79.90	76.19	0.00	34.99	42.41	0.01
conv (7)	91.94	83.76	79.86	11.96	0.00	9.50	42.42	10.10
bn (8)	92.46	81.24	79.93	66.10	0.00	36.86	42.51	0.01
conv (10)	92.11	83.72	79.96	14.79	0.00	12.76	42.45	7.13
bn (11)	92.53	80.14	79.88	10.00	0.00	37.79	42.42	10.00
conv (14)	92.07	82.87	80.00	14.95	0.00	16.87	42.11	1.23
bn (15)	92.46	79.74	79.92	10.00	0.00	37.88	42.36	10.00
conv (17)	92.11	81.94	80.08	15.27	0.00	20.73	41.84	0.00
bn (18)	92.56	79.55	79.94	10.00	0.00	35.86	42.45	10.00
conv (20)	92.10	81.18	80.29	27.76	0.00	24.81	41.32	0.00
bn (21)	92.39	79.79	79.94	10.02	0.00	34.75	42.38	10.00
conv (24)	92.39	80.67	80.22	22.88	0.00	28.09	40.69	0.16
bn (25)	92.52	80.50	80.06	10.04	0.00	35.34	42.31	10.03
conv (27)	92.47	80.67	80.48	29.50	0.00	29.93	39.77	0.02
bn (28)	92.54	80.24	79.97	10.00	0.00	36.32	42.35	10.01
conv (30)	92.56	80.19	80.25	34.58	0.00	30.00	41.17	0.29
bn (31)	92.53	80.50	80.01	42.78	0.00	36.91	42.36	9.43
conv (34)	92.57	80.63	79.91	51.30	0.00	33.08	42.15	0.25
bn (35)	92.55	80.86	79.88	41.99	0.00	38.33	42.30	9.99
conv (37)	92.53	81.12	79.88	40.24	0.00	36.15	42.52	3.11
bn (38)	92.53	80.89	79.89	10.73	0.00	38.68	42.60	3.17
conv (40)	92.57	80.98	79.89	35.76	0.00	37.25	42.53	0.35
bn (41)	92.53	80.86	79.82	19.65	0.00	39.25	42.46	8.78
linear	92.57	81.15	79.84	71.06	0.00	38.63	42.75	1.09