Explainable Adversarial Learning: Implicit Generative Modeling of Random Noise during Training for Adversarial Robustness

Anonymous authors
Paper under double-blind review

Abstract

We introduce Explainable Adversarial Learning, ExL, an approach for training neural networks that are intrinsically robust to adversarial attacks. We find that the implicit generative modeling of random noise with the same loss function used during posterior maximization, improves a model's understanding of the data manifold furthering adversarial robustness. We prove our approach's efficacy and provide a simplistic visualization tool for understanding adversarial data, using Principal Component Analysis. Our analysis reveals that adversarial robustness, in general, manifests in models with higher variance along the high-ranked principal components. We show that models learnt with our approach perform remarkably well against a wide-range of attacks. Furthermore, combining ExL with state-of-the-art adversarial training extends the robustness of a model, even beyond what it is adversarially trained for, in both white-box and black-box attack scenarios.

1 Introduction

Despite surpassing human performance on several perception tasks, Machine Learning (ML) models remain vulnerable to *adversarial examples*: slightly perturbed inputs that are specifically designed to fool a model during test time (Biggio et al., 2013; Szegedy et al., 2013; Goodfellow et al., 2014; Papernot et al., 2016a). Recent works have demonstrated the security danger adversarial attacks pose across several platforms with ML backend such as computer vision (Szegedy et al., 2013; Goodfellow et al., 2014; Moosavi Dezfooli et al., 2016; Kurakin et al., 2016; Liu et al., 2016), malware detectors (Laskov et al., 2014; Xu et al., 2016; Grosse et al., 2016; Hu & Tan, 2017) and gaming environments (Huang et al., 2017; Behzadan & Munir, 2017). Even worse, adversarial inputs *transfer* across models: same inputs are misclassified by different models trained for the same task, thus enabling simple *Black-Box* (BB) [1]attacks against deployed ML systems (Papernot et al., 2017).

Several works (Krotov & Hopfield, 2017; Papernot et al., 2016b; Cisse et al., 2017) demonstrating improved adversarial robustness have been shown to fail against stronger attacks (Athalye et al., 2018). The state-of-the-art approach for BB defense is ensemble adversarial training that augments the training dataset of the target model with adversarial examples transferred from other pre-trained models (Tramèr et al., 2017a). Madry et al. (2017) showed that models can even be made robust to *White-Box* (WB)[1] attacks by closely maximizing the model's loss with Projected Gradient Descent (PGD) based adversarial training. Despite this progress, errors still appear for perturbations beyond what the model is adversarially trained for (Sharma & Chen, 2017).

There have been several hypotheses explaining the susceptibility of ML models to such attacks. The most common one suggests that the overly linear behavior of deep neural models in a high dimensional input space causes adversarial examples (Goodfellow et al., 2014; Lou et al., 2016). Another hypothesis suggests that adversarial examples are off the data manifold (Goodfellow et al., 2016; Song et al., 2017; Lee et al., 2017). Combining the two, we infer that excessive linearity causes models to extrapolate their behavior beyond the data manifold yielding pathological results for slightly perturbed inputs. A question worth asking here is: *Can we improve the viability of the model to generalize better on such out-of-sample data?*

[1] BB (WB): attacker has no (full) knowledge of the target model parameters

In this paper, we propose *Explainable Adversarial Learning (ExL)*, wherein we introduce multiplicative noise into the training inputs and optimize it with Stochastic Gradient Descent (SGD) while minimizing the overall cost function over the training data. Essentially, the input noise (randomly initialized at the beginning) is gradually learnt during the training procedure. As a result, the noise approximately models the input distribution to effectively maximize the likelihood of the class labels given the inputs. Fig. 1 (a) shows the input noise learnt during different stages of training by a simple convolutional network ($ConvNet2$ architecture discussed in Section 3 below), learning handwritten digits from MNIST dataset (LeCun et al., 1998). We observe that the noise gradually transforms and finally assumes a shape that highlights the most dominant features in the MNIST training data. For instance, the MNIST images are centered digits on a black background. Noise, in fact, learnt this centered characteristic. This suggests that the model not only finds the *right prediction* but also the *right explanation*. Noise inculcates this explainable behavior by discovering some knowledge about the input/output distribution during training. Fig. 1 (b) shows the noise learnt with ExL on colored CIFAR10 images (Krizhevsky & Hinton, 2009) (on ResNet18 architecture (He et al., 2016)), which reveals that noise template (also RGB) learns prominent color blobs on a greyish-black background, that de-emphasizes background pixels.

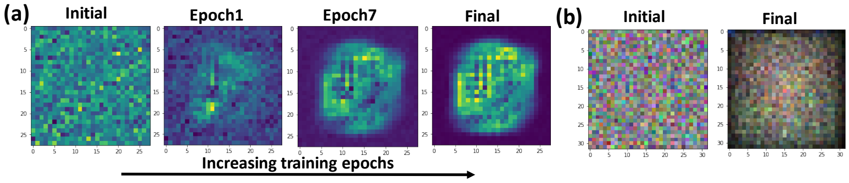


Figure 1: (a) Noise learnt with ExL on MNIST data- (b) Noise learnt with ExL on CIFAR10 data- with minibatch size =64.The template shown is the mean across all 64 noise templates.

A recent theory (Gilmer et al., 2018) suggests that adversarial examples (off manifold misclassified points) occur in close proximity to randomly chosen inputs on the data manifold that are, in fact, correctly classified. With ExL, we hypothesize that the model learns to look in the vicinity of the on-manifold data points and thereby incorporate more out-of-sample data (without using any direct data augmentation) that, in turn, improves its generalization capability in the off-manifold input space. We empirically evaluate this hypothesis by visualizing and studying the relationship between the adversarial and the clean inputs using Principal Component Analysis (PCA). Examining the intermediate layer's output, we discover that models exhibiting adversarial robustness yield significantly lower distance between adversarial and clean inputs in the Principal Component (PC) subspace.We further harness this result to establish that ExL noise modeling, indeed, acquires an improved realization of the input/output distribution characteristics that enables it to generalize better. To further substantiate our hypothesis, we also show that ExL globally reduces the dimensionality of the space of adversarial examples (Tramèr et al., 2017b). We evaluate our approach on classification tasks such as MNIST, CIFAR10 and CIFAR100 and show that models trained with ExL are extensively more adversarially robust. We also show that combining ExL with ensemble/PGD adversarial training significantly extends the robustness of a model, even beyond what it is adversarially trained for, in both BB/WB attack scenarios.

2 Explainable Learning

2.1 Approach

The basic idea of ExL is to inject random noise with the training data, continually minimizing the overall loss function by learning the parameters, as well as the noise at every step of training. The noise, N, dimensionality is same as the input, X, that is, for a $32 \times 32 \times 3$ sized image, the noise is $32 \times 32 \times 3$. In all our experiments, we use mini-batch SGD optimization. Let's assume the size of the training minibatch is m and the number of images in the minibatch is k, then, total training images are $m \times k$. Now, the total number of noisy templates are equal to the total number of inputs in each minibatch, k. Since, we want to learn the noise, we use the same k noise templates across all mini-batches $1, 2, ..., m$. This ensures that the noise templates inherit characteristics from the entire training dataset. Algorithm 1 shows the training procedure. It is evident from Algorithm 1 that noise learning at every training step follows the overall loss ($\mathcal{L}$, say cross-entropy) minimization that in turn enforces the maximum likelihood of the posterior.

Algorithm 1 Explainable Adversarial Learning of a model f with parameters θ, Loss Function $\mathcal{L}$.

```
Input: Input image $X$, Target label $Y$, Noise $N$, Learning rates $\eta$, $\eta_{noise}$
Output: Learnt noise $N$ and parameters $\theta$.
Randomly initialize the parameters $\theta$ and Noise $N : \{N^1, ...N^k\}$.
repeat
for each minibatch $\{X^{[1]}, ..., X^{[m]}\}$ do
    Input $X$= $\{X^1, ..., X^k\}$
    New input $X$= $\{X^1 \times N^1, ..., X^k \times N^k\}$
    Forward Propagation: $\hat{Y} = f(X; \theta)$
    Compute loss function: $\mathcal{L}(\hat{Y}, Y)$
    Backward Propagation: $\theta = \theta - \eta\nabla_\theta\mathcal{L}$; $N = N - \eta_{noise}\nabla_N\mathcal{L}$
end for
until training converges
```

Since adversarial attacks are created by adding perturbation to the clean input images, we were initially inclined toward using additive noise ($X + N$) instead of multiplicative noise ($X \times N$) to perform ExL. However, we found that ExL training with $X \times N$ tends to learn improved noise characteristics by the end of training. Fig. 2 (a) shows the performance results for different ExL training scenarios. While ExL with $X + N$ suffers a drastic $\sim 10\%$ accuracy loss with respect to standard SGD on clean data, $X \times N$ yields comparable accuracy. Furthermore, we observe that using only negative gradients (i.e. $\nabla_N\mathcal{L} \leq 0$) during backpropagation for ExL yields best accuracy (and closer to that of standard SGD trained model). Visualizing a sample image with learnt noise after training, in Fig. 2 (b), shows $X + N$ disturbs the original image severely, while $X \times N$ has a faint effect, corroborating the accuracy results. Since noise is modeled while conducting discriminative training, the multiplicative/additive nature of noise influences the overall optimization. Thus, we observe that noise templates learnt with $X \times N$ and $X + N$ are very different. We also analyzed the adversarial robustness of the models when subjected to WB attacks created using the Fast Gradient Sign Method (FGSM) for different perturbation levels (ϵ) (Fig. 2 (a)). ExL, for both $X \times N/X + N$ scenarios, yields improved accuracy than standard SGD. This establishes the effectiveness of the noise modeling technique during discriminative training towards improving a model's intrinsic adversarial resistance. Still, $X \times N$ yields slightly better resistance than $X + N$. Based upon these empirical studies, we chose to conform to multiplicative noise training in this paper. [2] Note, WB attacks, in case of ExL, are crafted using the model's parameters as well as the learnt noise N.

In all our experiments, we initialize the noise N from a random uniform distribution in the range $[0.8, 1]$. We select a high range in the beginning of training to limit the corruption induced on the training data due to the additional noise. During evaluation/testing, we take the mean of the learnt noise across all the templates ($(\sum_{i=1}^{k} N_i)/k$), multiply the averaged noise with each test image and feed it to the network to obtain the final prediction. Next, we present a general optimization perspective considering the maximum likelihood criterion for a classification task to explain adversarial robustness. It is worth mentioning that while Algorithm 1 describes the backpropagation step simply by using gradient updates, we can use other techniques like regularization, momentum etc. for improved optimization.

2.2 Adversarial Robustness from Likelihood Perspective

Given a data distribution D with inputs $X \in \mathbb{R}^d$ and corresponding labels Y, a classification/discriminative algorithm models the conditional distribution $p(Y|X; \theta)$ by learning the parameters θ. Since X inherits only the on-manifold data points, a standard model thereby becomes susceptible to adversarial attacks. For adversarial robustness, inclusion of the off-manifold data points while modeling the conditional probability is imperative. An adversarially robust model should, thus, model $p(Y|X, \mathbb{A}; \theta)$, where $\mathbb{A}$ represents the adversarial inputs. Using Bayes rule, we can derive the prediction obtained from posterior modeling from a generative standpoint as:

$$\underset{Y}{argmax}\, p(Y|X, \mathbb{A}) = \underset{Y}{argmax}\, \frac{p(\mathbb{A}|X, Y)p(X, Y)}{p(X, \mathbb{A})} = \underset{Y}{argmax}\, p(\mathbb{A}|X, Y)p(X|Y)p(Y) \qquad (1)$$

[2] Additional studies on other datasets comparing $X + N$ vs. $X \times N$ with different gradient update conditions can be found in Appendix A. See, experimental details and model description for Fig. 2 in Appendix C.1.

(a)

Scenario	Gradient $\nabla_N \mathcal{L}$	Clean	WhiteBox
X+N	Negative	78.1	59.1/49.6
	All	77.1	57.48/49.03
X×N	Negative	87.08	62.5/51.6
	All	85.1	62.4/51.9
SGD (X)	-	88.85	50.6/44.9

CIFAR10 Accuracy (in %) of ResNet18 target model for clean test data and test data perturbed with WB-FGSM attacks for ε=(8/16)/255 for different training scenarios (ExL, SGD).

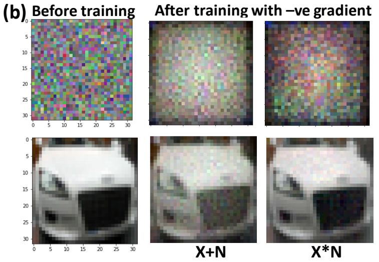


Figure 2: For multiplicative and additive noise training scenarios- (a) -accuracy comparison of ExL with SGD (b) -RGB noise template learnt with ExL on CIFAR10 data. In (b), a sample training image of a 'car' before and after training with noise is shown. Note, we used the same hyperparameters (batch-size =64, η, η_{noise} etc.) and same inital noise template across all scenarios during training. Noise shown is the mean across 64 templates.[2]

The methods employing adversarial training (Tramèr et al., 2017a; Kurakin et al., 2016; Madry et al., 2017) directly follow the left-hand side of Eqn. 1 wherein the training data is augmented with adversarial samples ($A \in \mathbb{A}$). Such methods showcase adversarial robustness against a particular form of adversary (e.g. ℓ_∞-norm bounded) and hence remain vulnerable to stronger attack scenarios. In an ideal case, A must encompass all set of adversarial examples (or the entire space of off-manifold data) for a concrete guarantee of robustness. However, it is infeasible to anticipate all forms of adversarial attacks during training. From a generative viewpoint (right-hand side of Eqn. 1), adversarial robustness requires modeling of the adversarial distribution while realizing the joint input/output distribution characteristics ($p(X|Y), p(Y)$). Yet, it remains a difficult engineering challenge to create rich generative models that can capture these distributions accurately. Some recent works leveraging a generative model for robustness use a PixelCNN model (Song et al., 2017) to detect adversarial examples, or use Generative Adversarial Networks (GANs) to generate adversarial examples (Samangouei et al., 2018). But, one might come across practical difficulties while implementing such methods due to the inherent training difficulty.

With Explainable Adversarial Learning, we partially address the above difficulty by modeling the noise based on the prediction loss of the posterior distribution. First, let us assume that the noise ($\mathbb{N}$) introduced with ExL spans a subspace of potential adversarial examples ($\mathbb{N} \subseteq \mathbb{A}$). Based on Eqn. 1 the posterior optimization criterion with noise ($\mathbb{N}$) becomes $argmax_Y\, p(Y|X, \mathbb{N}) = argmax_Y\, p(\mathbb{N}|X, Y)p(X|Y)p(Y)$. The noise learning in ExL (Algorithm 1) indicates an implicit generative modeling behavior, that is constrained towards maximizing $p(\mathbb{N}|X, Y)$ while increasing the likelihood of the posterior $p(Y|X, \mathbb{N})$. We believe that this partial and implicit generative modeling perspective with posterior maximization, during training, imparts an ExL model more knowledge about the data manifold, rendering it less susceptible toward adversarial attacks (See Appendix D for further intuition). Next, we empirically demonstrate using PCA that, noise modeling indeed embraces some off-manifold data points.

2.3 PC Subspace Analysis for Explainability & Visualization

PCA serves as a method to reduce a complex dataset to lower dimensions to reveal sometimes hidden, simplified structure that often underlie it. Since the learned representations of a deep learning model lie in a high dimensional geometry of the data manifold, we opted to reduce the dimensionality of the feature space and visualize the relationship between the adversarial and clean inputs in this reduced PC subspace. Essentially, we find the principal components (or eigen-vectors) of the activations of an intermediate layer of a trained model and project the learnt features onto the PC space. To do this, we center the learned features about zero ($\mathcal{F}$), factorize $\mathcal{F}$ using Singular Value Decomposition (SVD), i.e. $\mathcal{F} = USV^T$ and then transform the feature samples $\mathcal{F}$ onto the new subspace by computing $\mathcal{F}V = US \equiv \mathcal{F}^{PC}$. In Fig. 3 (a), we visualize the learnt representations of the *Conv1 layer* of a ResNet18 model trained on CIFAR-10 (with standard SGD) along different 2D-projections of the PC subspace in response to adversarial/clean input images. Interestingly, we see that the model's perception of both the adversarial and clean inputs along high-rank PCs (say, PC1- PC10 that account for maximum variance in the data) is alike. As we move toward lower-rank dimensions, the adversarial and clean image representations dissociate. This implies that adversarial images place strong emphasis on PCs that account for little variance in the data. While we note a similar trend with ExL (Fig. 3 (b)), the dissociation occurs at latter PC dimensions compared to Fig.

3 (a). A noteworthy observation here is that, adversarial examples lie in close vicinity of the clean inputs for both ExL/SGD scenarios ascertaining former theories of (Gilmer et al., 2018).

To quantify the dissociation of the adversarial and clean projections in the PC subspace, we calculate the cosine distance ($\mathcal{D}^{PC} = \frac{1}{N}\sum_{i=1}^{N} 1 - \frac{\mathcal{F}^{PC}_{clean\,i} \cdot \mathcal{F}^{PC}_{adv\,i}}{\|\mathcal{F}^{PC}_{clean\,i}\|_2 \|\mathcal{F}^{PC}_{adv\,i}\|_2}$) between them along different PC dimensions. Here, N represents the total number of sample images used to perform PCA and $F^{PC}_{clean}(F^{PC}_{adv})$ denote the transformed learnt representations corresponding to clean (adversarial) input, respectively. The distance between the learnt representations (for the *Conv1 layer* of ResNet18 model from the above scenario) consistently increases for latter PCs as shown in Fig. 3 (c). Interestingly, the cosine distance between adversarial and clean features measured for a model trained with ExL noise is significantly lesser than a standard SGD trained model. This indicates that noise enables the model to look in the vicinity of the original data point and inculcate more adversarial data into its underlying representation. Note, we consider projection across all former dimensions (say, PC0, PC1,...PC100) to calculate the distance at a later dimension (say, PC100) i.e., ${\mathcal{D}^{PC}}_{100}$ is calculated by taking the dot product between two 100-dimensional vectors: $\mathcal{F}^{PC}_{clean}, \mathcal{F}^{PC}_{adv}$.

To further understand the role of ExL noise in a model's behavior, we analyzed the variance captured in the *Conv1* layer's activations of the ResNet18 model (in response to clean inputs) by different PCs, as illustrated Fig. 3 (d). If $s_i = \{1, ..., M\}$ are the singular values of the matrix S, the variance along a particular dimension PC_k is defined as: $Var_k = 100 \times (\sum_{i=0}^{k} {s_i}^2 / \sum_{i=0}^{M} {s_i}^2)$. Var_k along different PCs provides a good measure of how much a particular dimension explains about the data. We observe that ExL noise increases the explainability (or variance) along the high rank PCs, for instance, the net variance obtained from PC0-PC100 with ExL Noise (90%) is more than that of standard SGD (76%). In fact, we observe a similar increase in variance in the leading PC dimensions for other intermediate blocks learnt activations of the ResNet18 model [See *Appendix B*]. We can infer that the increase in variance along the high-rank PCs is a consequence of inclusion of more data points during the overall learning process. Conversely, we can also interpret this as ExL noise embracing more off-manifold adversarial points into the overall data manifold that eventually determines the model's behavior. It is worth mentioning that the variance analysis of the model's behavior in response to adversarial inputs yields nearly identical results as Fig. 3 (d) [*Appendix B*].

Interestingly, the authors in (Hendrycks & Gimpel, 2017) conducted PCA whitening of the raw image data for clean and adversarial inputs and demonstrated that adversarial image coefficients for later PCs have greater variance. Our results from PC subspace analysis corroborates their experiments and further enables us to peek into the model's behavior for adversarial attacks. Note, for all the PCA experiments above, we used 700 random images sampled from the CIFAR-10 test data, i.e. $N = 700$. In addition, we used the Fast Gradient Sign Method (FGSM) method to create BB adversaries with a step size of $8/255$, from a different source model (ResNet18 trained with SGD).

3 Results

3.1 Attack Methods

Given a test image X, an attack model perturbs the image to yield an adversarial image, $X_{adv} = X + \Delta$, such that a classifier f misclassifies X_{adv}. In this work, we consider ℓ_∞ bounded adversaries studied in earlier works (Goodfellow et al., 2014; Tramèr et al., 2017a; Madry et al., 2017), wherein the perturbation ($\|\Delta\|_\infty \leq \epsilon$) is regulated by some parameter ϵ. Also, we study robustness against both BB/WB attacks to gauge the effectiveness of our approach. For an exhaustive assessment, we consider the same attack methods deployed in Tramèr et al. (2017a); Madry et al. (2017):

Fast Gradient Sign Method (FGSM): This single-step attack is a simple way to generate malicious perturbations in the direction of the loss gradient $\nabla_X \mathcal{L}(X, Y_{true})$ as: $X_{adv} = X + \epsilon sign(\nabla_X \mathcal{L}(X, Y_{true}))$.

Random Step FGSM (R-FGSM): (Tramèr et al., 2017a) suggested to prepend single-step attacks with a small random step to escape the non-smooth vicinity of a data point that might degrade attacks based on single-step gradient computation. For parameters ϵ, α ($\alpha = \epsilon/2$), the attack is defined as: $X_{adv} = X + \epsilon sign(\nabla_X \mathcal{L}(X, Y_{true})),\ where\ X = X + \alpha sign(\mathcal{N}(0^d, I^d))$.

Iterative FGSM (I-FGSM): This method iteratively applies FGSM k times with a step size of $\beta \geq \epsilon/k$ and projects each step perturbation to be bounded by ϵ. Following (Tramèr et al., 2017a),

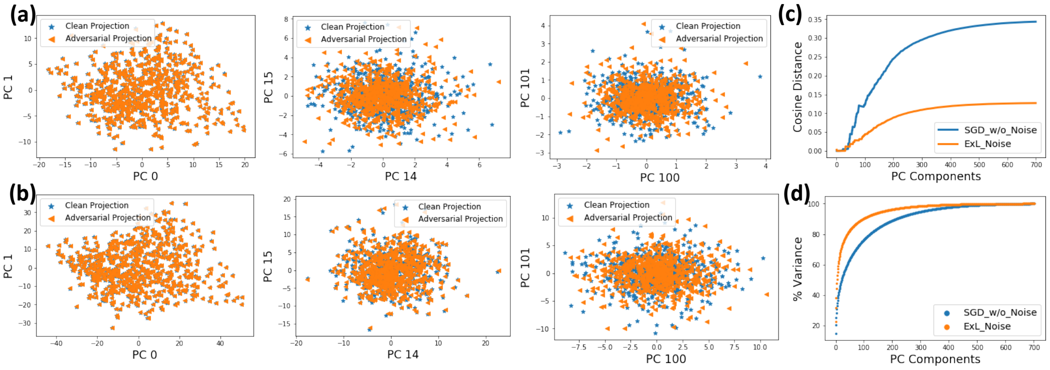


Figure 3: Relationship between the model's understanding of adversarial and clean inputs in PC subspace when trained with (a) SGD (b) ExL. (c) Cosine Distance between the model's response to clean and adversarial inputs in the PC subspace. (d) Variance of the $Conv1$ layer of ResNet18 model. (c), (d) compare the SGD/ ExL training scenarios.

we use two-step iterative FGSM attacks.

Projected Gradient Descent (PGD): Similar to I-FGSM, this is a multi-step variant of FGSM: $X_{adv}{}^{t+1} = \prod(X_{adv}{}^{t} + \alpha sign(\mathcal{L}(X, Y_{true})))$. Madry et al. (2017) show that this is a universal first-order adversary created by initializing the search for an adversary at a random point followed by several iterations of FGSM. PGD attacks, till date, are one of the strongest BB/ WB adversaries.

3.2 Experiments

We evaluated ExL on three datasets: MNIST, CIFAR10 and CIFAR100. For each dataset, we report the accuracy of the models against BB/WB attacks (crafted from the test data) for 6 training scenarios: a) Standard SGD (without noise), b) ExL Noise, c) Ensemble Adversarial (EnsAdv) Training (SGD_{ens}), d) ExL Noise with EnsAdv Training (ExL_{ens}), e) PGD Adversarial (PGDAdv) Training (SGD_{PGD}), f) ExL Noise with PGDAdv Training (ExL_{PGD}). Note, SGD_{ens} and SGD_{PGD} refer to the standard adversarial training employed in Tramèr et al. (2017a) and Madry et al. (2017), respectively. Our results compare how the additional noise modeling improves over standard SGD in adversarial susceptibility. Also, we integrate ExL with state-of-the-art PGD/Ensemble adversarial training techniques to analyze how noise modeling benefits them. In case of EnsAdv training, we augmented the training dataset of the target model with adversarial examples (generated using R-FGSM), from an independently trained model, with same architecture as the target model. In case of PGDAdv training, we augmented the training dataset of the target model with adversarial examples (generated using PGD) from the same target model. Thus, as we see later, EnsAdv imparts robustness against BB attacks only, while, PGD makes a model robust to both BB/WB attacks. In all experiments below, we report the WB/BB accuracy against strong adversaries created with PGD attack. In additon, for BB, we also report the worst-case error over all small-step attacks FGSM, I-FGSM, R-FGSM, denoted as *Min BB* in Table 1, 2.

All networks were trained with mini-batch SGD using a batch size of 64 and momentum of 0.9 (0.5) for CIFAR (MNIST), respectively. For CIFAR10, CIFAR100 we used additional weight decay regularization, $\lambda = 5e-4$. Note, for noise modeling, we simply used the negative loss gradients ($\nabla_N \mathcal{L} \leq 0$) without additional optimization terms. In general, ExL requires slightly more epochs of training to converge to similar accuracy as standard SGD, a result of the additional input noise modeling. Also, ExL models, if not tuned with proper learning rate, have a tendency to overfit. Hence, the learning rate for noise (η_{noise}) was kept 1-2 orders of magnitude lesser than the overall network learning rate (η) throughout the training process. All networks were implemented in PyTorch.[3]

MNIST: For MNIST, we consider a simple network with 2 Convolutional (C) layers with 32, 64 filters, each followed by 2×2 Max-pooling (M), and finally a Fully-Connected (FC) layer of size 1024, as the target model (ConvNet1: 32C-M-64C-M-1024FC). We trained 6 ConvNet1 models independently corresponding to the different scenarios. The EnsAdv (ExL_{ens}, SGD_{ens}) models

[3] *Appendix C* provides a detailed table of different hyperparameters used to train the source and target models in each scenario corresponding to all experiments of Table 1, 2 . Appendix C shows different visualization of noise learnt (N) in each scenario of Table 1, 2 . Note, the code for noise modeling and corresponding attack scenarios will be available in the url: [link omitted for anonymity].

were trained with BB adversaries created from a separate SGD-trained ConvNet1 model using R-FGSM with $\epsilon = 0.1$. PGDAdv (ExL_{PGD}, SGD_{PGD}) models were trained with WB adversaries created from the same target model using PGD with $\epsilon = 0.3$, step-size = 0.01 over 40 steps.

Table 1 (Columns 3 - 5) illustrates our results for BB attacks under different perturbations (ϵ)[4]. ExL noise considerably improves the robustness of a model toward BB attacks. An interesting observation here is that for $\epsilon = 0.1$ (that was the perturbation size for EnsAdv training), both ExL_{ens}/SGD_{ens} yield nearly similar accuracy, $\sim 98\%$. However, for larger perturbation size $\epsilon = 0.2, 0.3$, the network adversarially trained with ExL noise shows higher prediction capability ($\sim > 5\%$) across the PGD attack methods. We observe similar BB accuracy trend with PGDAdv methods (ExL_{PGD}/SGD_{PGD}). Columns 6-7 in Table 1 show the WB attack results. All techniques except for the ones with PGDAdv training fail miserably against the strong WB PGD attacks. Models trained with ExL noise, although yielding low accuracy, still perform better than SGD. ExL_{PGD} yields better accuracy than SGD_{PGD} even beyond what the network is adversarially trained for ($\epsilon > 0.3$). Note, for PGD attack in Table 1, we used a step-size of 0.01 over 40/100 steps to create adversaries bounded by $\epsilon = 0.1/0.2/0.3$. We also evaluated the worst-case accuracy over all the BB attack methods when the source model is trained with ExL noise (not shown). We found higher accuracies in this case, implying ExL models *transfer* attacks at lower rates. As a result, in the remainder of the paper, we conduct BB attacks from models trained without noise modeling to evaluate the adversarial robustness[4].

Table 1: **MNIST Accuracy (in %) of ConvNet1 target model for different scenarios.** $\epsilon = 0.1/0.2/0.3$ for $SGD, ExL, SGD_{ens}, ExL_{ens}$; $\epsilon = 0.3/0.4$ for SGD_{PGD}, ExL_{PGD}. For PGD attack, we report accuracy for 40-/100-step attacks. Note, SGD_{PGD}, ExL_{PGD} have stronger BB attacks than remaining scenarios[4]. Accuracy $< 5\%$, in most places, have been omitted and marked as '-'.

Scenario	Clean	Min BB	PGD-40	PGD-100	PGD-40	PGD-100
		(————**BlackBox**————		)	(——**WhiteBox**——	)
SGD	99.1	77.9/20.6/4.3	75/9.9/-	74.5/8/-	22.3/-/-	-
ExL	99.2	83.6/30.5/9.6	80.5/20.6/-	80/18/-	29.4/-/-	-
SGD_{ens}	99	98.5/92.6/73.2	98/89.3/71	98.1/88/57	2.1/-/-	-
ExL_{ens}	99.1	99/94.7/76	98.8/93.4/79	98.7/91.9/66	3.3/-/-	-
SGD_{PGD}	97.9	91.8/29	93.6/48.7	92.3/20	90/27	86.5/4.5
ExL_{PGD}	98	93/42.2	94/60.4	92.6/28.7	90.7/55.7	88/20.1

CIFAR: For CIFAR10, we examined our approach on the ResNet18 architecture. We used the ResNext29(2×64d) architecture (Xie et al., 2017) with bottleneck width 64, cardinality 2 for CIFAR100. Similar to MNIST, we trained the target models separately corresponding to each scenario and crafted BB/WB attacks. For EnsAdv training, we used BB adversaries created using R-FGSM ($\epsilon = 8/255$) from a separate SGD-trained network different from the BB source/target model. For PGDAdv training, the target models were trained with WB adversaries created with PGD with $\epsilon = 8/255$, step-size=$2/255$ over 7 steps. Here, for PGD attacks, we use 7/20 steps of size $2/255$ bounded by ϵ. The results appear in Table 2.

For BB, we observe that ExL ($81\%/63.2\%$ for CIFAR10/100) significantly boosts the robustness of a model as compared to SGD ($50.3\%/44.2\%$ for CIFAR10/100). Note, the improvement here is quite large in comparison to MNIST (that shows only 5% increase from SGD to ExL). In fact, the accuracy obtained with ExL alone with BB attack, is almost comparable to that of an EnsAdv trained model without noise (SGD_{ens}). The richness of the data manifold and feature representation space for larger models and complex datasets allows ExL to model better characteristics in the noise causing increased robustness. As seen earlier, ExL noise (ExL_{ens}, ExL_{PGD}) considerably improves the accuracy even for perturbations ($\epsilon = (16, 32)/255$) greater than what the network is adversarially trained for. The increased susceptibility of SGD_{ens}, SGD_{PGD} for larger ϵ establishes that its capability is limited by the diversity of adversarial examples shown during training. For WB attacks as well, ExL_{PGD} show higher resistance. Interestingly, while SGD, SGD_{ens} yield infinitesimal performance ($< 5\%$), ExL, ExL_{ens} yield reasonably higher accuracy ($> 25\%$) against

[4] For fair comparison, BB attacks on $SGD, ExL, SGD_{ens}, ExL_{ens}$ were crafted from another model trained with standard SGD on natural examples as in (Tramèr et al., 2017a). While, BB attacks on SGD_{PGD}, ExL_{PGD} were crafted from a model trained with PGDAdv training (without noise modeling) on adversarial examples as in (Madry et al., 2017) to cast stronger attacks.

WB attacks. This further establishes the potential of noise modeling in enabling adversarial security. It is worth mentioning that BB accuracy of SGD_{PGD}, ExL_{PGD} models in Table 1, 2 are lower than SGD_{ens}, ExL_{ens}, since the former is attacked with stronger attacks crafted from models trained with PGDAdv[4]. Attacking the former with similar adversaries as latter yields higher accuracy.

Table 2: **CIFAR10/ CIFAR100 Accuracy (in %) of ResNet18/ ResNext-29 target model for different scenarios.** $\epsilon = \frac{8}{255}/\frac{16}{255}/\frac{32}{255}$ for $ExL, SGD, ExL_{ens}, SGD_{ens}, ExL_{PGD}, SGD_{PGD}$. For PGD attack, we report accuracy for 7-/20-step attacks. Note, SGD_{PGD}, ExL_{PGD} have stronger BB attacks than remaining scenarios[4]. Accuracy $< 5\%$, in most places, have been omitted and marked as '-'.

Scenario	Clean	Min BB	PGD-7	PGD-20	PGD-7	PGD-20
		(—————	**BlackBox**	—————)	(——**WhiteBox**——)	
			ResNet18 (CIFAR10)			
SGD	88.8	50.3/32/16.2	34/23/17.1	28/11.2/6.2	2.1/-/-	-
ExL	87.1	81/76/67	80.1/75.7/66.4	80/74.8/61	39.1/29.2/23	-
SGD_{ens}	86.3	81.3/76.6/68.3	80.9/75.1/67.1	80.2/74.4/63	0.8/-/-	-
ExL_{ens}	86.4	84.4/81.4/72.6	83/80/71.3	82.7/79/71	29/21/16.5	-
SGD_{PGD}	83.2	71.3/58/50	69.9/62/50.1	54.2/50.3/46	58.4/48/42	57.3/42.8/28
ExL_{PGD}	83	73/62/56.8	71/65/53	57.6/53.7/49.8	63/59/57	59.2/45/30.1
			ResNext29 (CIFAR100)			
SGD	71	44.2/38.4/26.7	42.7/35/25.4	40.5/27/17	-	-
ExL	69.4	63.2/58.5/50.1	62.9/54.3/48.4	62.3/53.1/42.5	19/14/10.3	-
SGD_{ens}	69.8	64.8/60.9/50	63.6/57.5/45.4	63/56/42	2.5/-/-	-
ExL_{ens}	67.3	65.1/62.8/57	64.8/61.4/52.2	64.4/58/49	18/14/11	-
SGD_{PGD}	71.6	57.5/48/38.4	56/45/41.3	48/40/38.4	51.5/49.8/46	50.4/43/33
ExL_{PGD}	69	66.3/62/59.9	63/58.7/54.1	52.3/50/40.8	58.1/56/53	53/48/37.9

PC Distance & Variance Analysis : Next, we measured the variance and cosine distance captured by the $Conv1$ layer of the ResNet18 model corresponding to different scenarios (Table 2). Fig. 4 (a) shows that variance across the leading PCs decreases as $ExL_{PGD} > SGD_{PGD} > ExL_{ens} > ExL > SGD_{ens} > SGD$. Inclusion of adversarial data points with adversarial training or noise modeling informs a model more, leading to improved explainability. We note that ExL_{ens} and SGD_{PGD} yield nearly similar variance ratio, although SGD_{PGD} gives better accuracy than ExL_{ens} for similar BB and WB attacks. Since we are analyzing only the $Conv1$ layer, we get this result. In Fig. 4 (a), we also plot the cosine distance between the adversarial (created from FGSM with specified ϵ) and clean inputs in the PC subspace. The distance across different scenarios along latter PCs increases as: $ExL_{PGD} < SGD_{PGD} < ExL_{ens} < ExL < SGD_{ens} < SGD$. A noteworthy observation here is, PC distance follows the same order as decreasing variance and justifies the accuracy results in Table 2. The decreasing distance with ExL compared to SGD further signifies improved realization of the on-/off-manifold data. Also, the fact that ExL_{PGD}, ExL_{ens} have lower distance for varying ϵ establishes that integrating noise modeling with adversarial training compounds adversarial robustness. Interestingly, for both variance and PC distance, ExL has a better characteristic than SGD_{ens}. This proves that noise modeling enables implicit inclusion of adversarial data without direct data augmentation, as opposed to EnsAdv training (or SGD_{ens}) where the dataset is explicitly augmented. This also explains the comparable BB accuracy between ExL, SGD_{ens} in Table 2.

Adversarial Subspace Dimensionality : To further corroborate that ExL noise implicitly embraces adversarial points, we evaluated the adversarial subspace dimension using the Gradient-Aligned Adversarial Subspace (GAAS) method of (Tramèr et al., 2017b). We construct k orthogonal vectors $r_1, .., r_k \in \{-1, 1\}$ from a regular Hadamard matrix of order $k \in \{2^2, 2^3, .., 2^7\}$. We then multiply each r_i component-wise with the gradient, $sign(\nabla_X \mathcal{L}(X, Y_{true}))$. Hence, estimating the dimensionality reduces to finding a set of orthogonal perturbations, r_i with $\|r_i\|_\infty = \epsilon$ in the vicinity of a data point that causes misclassification. For each scenario of Table 2 (CIFAR10), we select 350 random test points, x, and plot the probability that we find at least k orthogonal vectors r_i such that $x + r_i$ is misclassified. Fig. 4 (b), (c) shows the results with varying ϵ for BB, WB instances. We find that the size of the space of adversarial samples is much lower for a model trained with ExL noise than that of standard SGD. For $\epsilon = 8/255$, we find over 128/64 directions for $\sim 25\%/15\%$ of the points in case of SGD/ExL. With EnsAdv training, the number of adversarial directions for

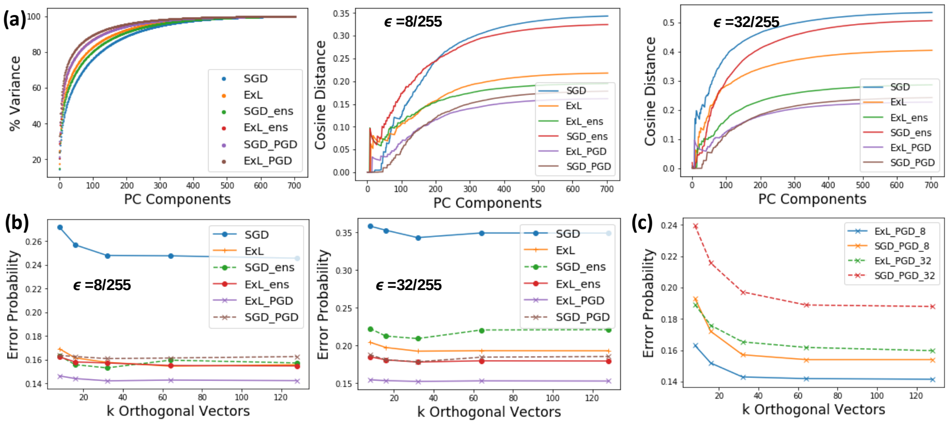


Figure 4: (a) [Left] Variance (in response to clean inputs) across different scenarios for the first 700 PC dimensions. [Middle, Right] Cosine distance across 700 PCs between clean and adversarial representations for varying ϵ. Adversarial subspace dimensionality for varying ϵ for- (b) -BB adversaries crafted from a model trained with natural examples (c) -WB adversaries crafted for models trained with PGDAdv training. SGD_{ens}, SGD_{PGD} **exhibit improved variance (and lower distance) than** SGD**, suggesting PC variance/ distance as a good indicator of adversarial robustness.** PCA was conducted with sample of 700 test images.

SGD_{ens}/ExL_{ens} reduces to 64 that misclassifies $\sim 17/15\%$ of the points. With PGDAdv training, the adversarial dimension significantly reduces in case of ExL_{PGD} for both BB/WB. As we increase the perturbation size ($\epsilon = 32/255$), we observe increasingly reduced number of misclassified points as well as adversarial dimensions for models trained with noise modeling. The WB adversarial plot, in Fig. 4 (c), clearly shows the reduced space obtained with noise modeling with PGDAdv training (ExL_{PGD}) against plain PGDAdv (SGD_{PGD}) for $\epsilon = (8, 32)/255$.

Loss Surface Smoothening: By now, it is clear that while ExL alone can defend against BB attacks (as compared to SGD) reasonably well, it still remains vulnerable to WB attacks. For WB defense and to further improve BB defense, we need to combine ExL noise modeling with adversarial training. To further investigate this, we plotted the loss surface of MNIST models on examples $x = x + \epsilon_1 \cdot g_{BB} + \epsilon_2 \cdot g_{WB}$ in Fig. 5, where g_{BB} is the signed gradient, $sign(\nabla_X \mathcal{L}(X, Y_{true})_{source})$, obtained from the source model (crafting the BB attacks) and g_{WB} is the gradient obtained from the target model itself (crafting WB attacks), $sign(\nabla_X \mathcal{L}(X, Y_{true})_{target})$. We see that the loss surface in case of SGD is highly curved with steep slopes in the vicinity of the data point in both BB and WB direction. The EnsAdv training, SGD_{ens}, smoothens out the slope in the BB direction substantially, justifying their robustness against BB attacks. Models trained with noise modeling, ExL (even without any data augmentation), yield a softer loss surface. This is why ExL models *transfer* BB attacks at lower rates. The surface in the WB direction along ϵ_2 with ExL, ExL_{ens} still exhibits a sharper curvature (although slightly softer than SGD_{ens}) validating the lower accuracies against WB attacks (compared to BB attacks). PGDAdv, on the other hand, smoothens out the loss surface substantially in both directions owing to the explicit inclusion of WB adversaries during training. Note, ExL_{PGD} yields a slightly softer surface than SGD_{PGD} (not shown). The smoothening effect of noise modeling further justifies the boosted robustness of ExL models for larger perturbations (outside ϵ-ball used during adversarial training). It is worth mentioning that we get similar PCA/ Adversarial dimensionality/ loss surface results across all datasets.

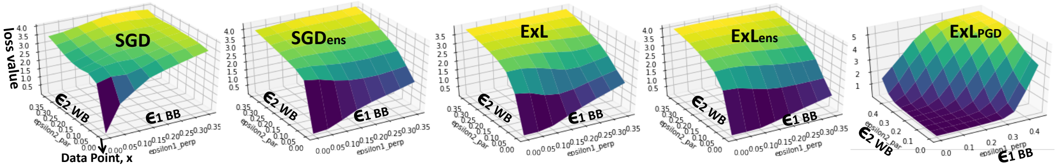


Figure 5: Loss surface of models corresponding to MNIST (Table1).

4 Discussion

We proposed *Explainable Adversarial Learning, ExL,* as a reliable method for improving adversarial robustness. Specifically, our key findings are:

1) We show that noise modeling at the input during discriminative training improves a model's ability to generalize better for out-of-sample adversarial data (without explicit data augmentation).
2) Our PCA variance and cosine distance analysis provides a significant perspective to visualize and quantify a model's response to adversarial/clean data.

A crucial question one can ask is, ***How to break ExL defense?*** The recent work (Athalye et al., 2018) shows that many defense methods cause 'gradient masking' that eventually fail. We reiterate that, ExL alone does not give a strong BB/WB defense. However, the smoothening effect of noise modeling on the loss (Fig. 5) suggests that noise modeling decreases the magnitude of the gradient masking effect. ExL does not change the classification model that makes it easy to be scaled to larger datasets while integrating with other adversarial defense techniques. Coupled with other defense, ExL performs remarkably (even for larger ϵ values). We combine ExL with EnsAdv & PGDAdv, which do not cause obfuscated gradients and hence can withstand strong attacks, however, upto a certain point. For WB perturbations much greater than the training ϵ value, ExL+PGDAdv also breaks. In fact, for adaptive BB adversaries Tramèr et al. (2017a) or adversaries that query the model to yield full prediction confidence (not just the label), ExL+EnsAdv will be vulnerable. Note, advantage with ExL is, being independent of the attack/defense method, ExL can be potentially combined with stronger attacks developed in future, to create stronger defenses.

While variance and principal subspace analysis help us understand a model's behavior, we cannot fully describe the structure of the manifold learnt by the linear subspace view. However, PCA does provide a basic intuition about the generalization capability of complex image models. In fact, our PC results establish the superiority of adversarial training methods ($SGD_{ens}; SGD_{PGD}$: Tramèr et al. (2017a); Madry et al. (2017) and can be used as a valid metric to gauge adversarial susceptibility in future proposals. Finally, as our likelihood theory (Eqn.1) indicates, better noise modeling techniques with improved gradient penalties can further improve robustness and requires further investigation. Also, performing noise modeling at intermediate layers to improve variance/explainability, and hence robustness, are other future work directions.

References

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pp. 262–275. Springer, 2017.

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

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

Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.

URL Github. `https://github.com/kuangliu/pytorch-cifar/tree/master/models`.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning.(2016). *Book in preparation for MIT Press. URL: http://www. deeplearningbook. org*, 2016.

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

Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial perturbations against deep neural networks for malware classification. *arXiv preprint arXiv:1606.04435*, 2016.

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*, pp. 770–778, 2016.

Dan Hendrycks and Kevin Gimpel. Early methods for detecting adversarial images. 2017.

Weiwei Hu and Ying Tan. Generating adversarial malware examples for black-box attacks based on gan. *arXiv preprint arXiv:1702.05983*, 2017.

Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018.

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

Dmitry Krotov and John J Hopfield. Dense associative memory is robust to adversarial inputs. *arXiv preprint arXiv:1701.00939*, 2017.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Pavel Laskov et al. Practical evasion of a learning-based classifier: A case study. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pp. 197–211. IEEE, 2014.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Hyeungill Lee, Sungyeob Han, and Jungwoo Lee. Generative adversarial trainer: Defense to adversarial perturbations with gan. *arXiv preprint arXiv:1705.03387*, 2017.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

Yan Lou, Xavier Boix, Gemma Roig, Tomaso Poggio, and Qi Zhao. Foveation-based mechanisms alleviate adversarial examples. Technical report, Center for Brains, Minds and Machines (CBMM), arXiv, 2016.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Seyed Mohsen Moosavi Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, number EPFL-CONF-218057, 2016.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387. IEEE, 2016a.

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*, pp. 582–597. IEEE, 2016b.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506–519. ACM, 2017.

Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, volume 9, 2018.

Yash Sharma and Pin-Yu Chen. Breaking the madry defense model with l_1-based adversarial examples. *arXiv preprint arXiv:1710.10733*, 2017.

Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

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.

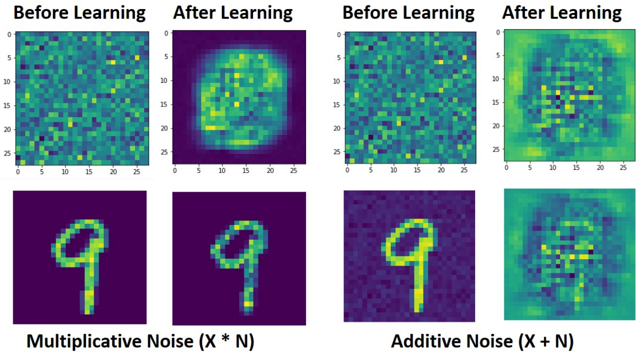
Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017a.

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

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 5987–5995. IEEE, 2017.

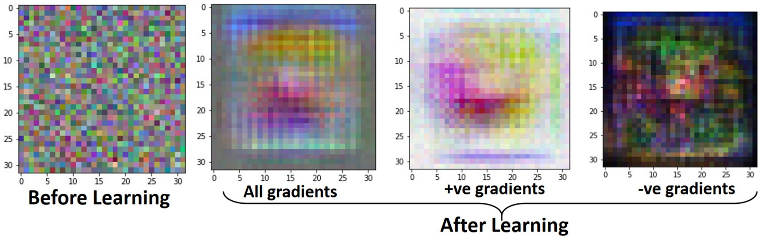
Weilin Xu, Yanjun Qi, and David Evans. Automatically evading classifiers. In *Proceedings of the 2016 Network and Distributed Systems Symposium*, 2016.

A Appendix A: Justification of $X + N$ vs $X \times N$ and use of $\nabla\mathcal{L}_N \leq 0$ for noise modeling



Scenario	Accuracy
SGD without noise	98.13%
ExL with noise (X * N)	98.62%
ExL with noise (X + N)	95.42%

Figure A1: For MNIST dataset, we show the noise template learnt when we use multiplicative/additive noise (N) for Explainable Learning. The final noise-integrated image (for a sample digit '9') that is fed to the network before and after training is also shown. Additive noise disrupts the image drastically. Multiplicative noise, on the other hand, enhances the relevant pixels while eliminating the background. Accuracy corrsponding to each scenario is also shown and compared against standard SGD training scenario (without any noise). Here, we train a simple convolutional architecture (ConvNet: 10C-M-20C-M-320FC) of 2 Convolutional (C) layers with 10, 20 filters, each followed by 2×2 Max-pooling (M) and a Fully-Connected (FC) layer of size 320. We use mini-batch SGD with momentum of 0.5, learning rate (η=0.1) decayed by 0.1 every 15 epochs and batch-size 64 to learn the network parameters. We trained 3 ConvNet models independently corresponding to each scenario for 30 epochs. For the ExL scenarios, we conduct noise modelling with only negative loss gradients ($\nabla\mathcal{L}_N \leq 0$) with noise learning rate, $\eta_{noise} = 0.001$, throughout the training process. Note, the noise image shown is the average across all 64 noise templates.



Scenario	Accuracy
SGD without noise	64.43%
ExL (all gradient)	59.94%
ExL (+ve gradient)	60.06%
ExL (-ve gradient)	63.45%

Figure A2: Here, we showcase the noise learnt by a simple convolutional network (ConvNet: 10C-M-20C-M-320FC), learning the CIFAR10 data with ExL (multiplicative noise) under different gradient update conditions. As with MNIST (Fig. A1), we observe that the noise learnt enhances the region of interest while deemphasizing the background pixels. Note, the noise in this case has RGB components as a result of which we see some prominent color blobs in the noise template after training. The performance table shows that using only negative gradients (i.e. $\nabla\mathcal{L}_N \leq 0$) during backpropagation for noise modelling yields minimal loss in accuracy as compared to a standard SGD trained model. We use mini-batch SGD with momentum of 0.9, weight decay 5e-4, learning rate (η=0.01) decayed by 0.2 every 10 epochs and batch-size 64 to learn the network parameters. We trained 4 ConvNet models independently corresponding to each scenario for 30 epochs. For the ExL scenarios, we conduct noise modelling by backpropagating the corresponding gradient with noise learning rate ($\eta_{noise} = 0.001$) throughout the training process. Note, the noise image shown is the average across all 64 noise templates.

B Appendix B: PC variance for SGD and ExL scenarios in response to adversarial and clean inputs across different layers of ResNet18

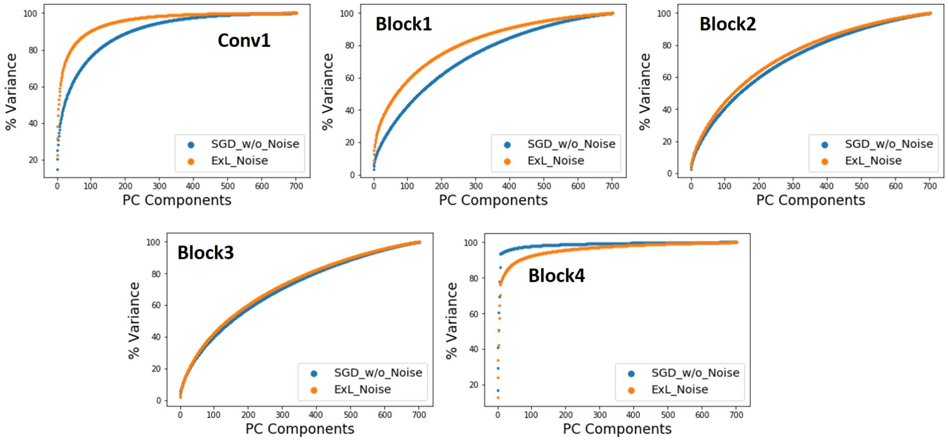


Figure A3: Here, we show the variance captured in the leading Principal Component (PC) dimensions for the inital convolutional layer's ($Conv1$) and intermediate blocks learnt activations of a ResNet-18 model trained on CIFAR10 data. We compare the variance of the learnt representations (in response to clean inputs) for each block across two scenarios: SGD (without noise) and ExL (with noise). Note, we capture the variance of the final block's activations before average pooling. That is, the activations of $Block4$ have dimension $512 \times 4 \times 4$. We observe that ExL noise increases the explainability (or variance) along the high rank PCs. Also, as we go deeper into the network, the absolute difference of the variance values between SGD/ExL decreases. This is expected as the contribution of input noise on the overall representations decreases as we go deeper into the network. Moreover, there is a generic-to-specific transition in the hierarchy of learnt features of a deep neural network. Thus, the linear PC subspace analysis to quantify a model's knowledge of the data manifold is more applicable in the earlier layers, since they learn more general input-related characteristics. Nonetheless, we see that ExL model yields widened explainability than SGD for each intermediate layer except the final $Block4$ that feeds into the output layer. We use mini-batch SGD with momentum of 0.9, weight decay 5e-4, learning rate (η=0.1) decayed by 0.1 every 30 epochs and batch-size 64 to learn the network parameters. We trained 2 ResNet-18 models independently corresponding to each scenario for 60 epochs. For noise modelling, we use $\eta_{noise} = 0.001$ decayed by 0.1 every 30 epochs. Note, we used a sample set of 700 test images to conduct the PCA.

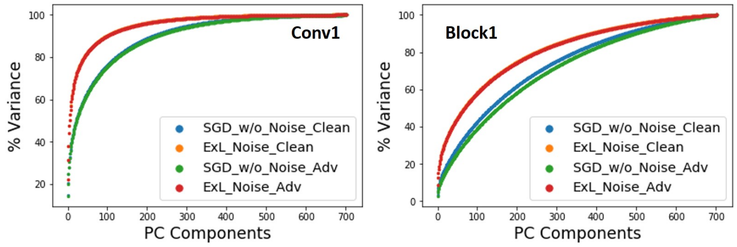


Figure A4: Here, we show the variance captured in the leading Principal Component (PC) dimensions for the $Conv1$ and $Block1$ learnt activations in response to both clean and adversarial inputs for ResNet-18 models correponding to the scenarios discussed in Fig. A3. The model's variance for both clean and adversarial inputs are exactly same in case of ExL/SGD for $Conv1$ layers. For $Block1$, the adversarial input variance is slighlty lower in case of SGD than that of clean input. With ExL, the variance is still the same for $Block1$. This indicates that PC variance statistics cannot differentiate between a model's knowledge of on-/off- manifold data. It only tells us whether a model's underlying representation has acquired more knowledge about the data manifold. To analyze a model's understanding of adversarial data, we need to look into the relationship between the clean and adversarial projection onto the PC subspace and measure the cosine distance. Note, we used the Fast Gradient Sign Method (FGSM) method Goodfellow et al. (2014) to create BB adversaries with a step size of $8/255$, from another independently trained ResNet-18 model ($source$) with standard SGD. The $source$ attack model has the same hyperparameters as the SGD model in Fig. A3 and is trained for 40 epochs.

C Appendix C: Experimental Details and Model Description

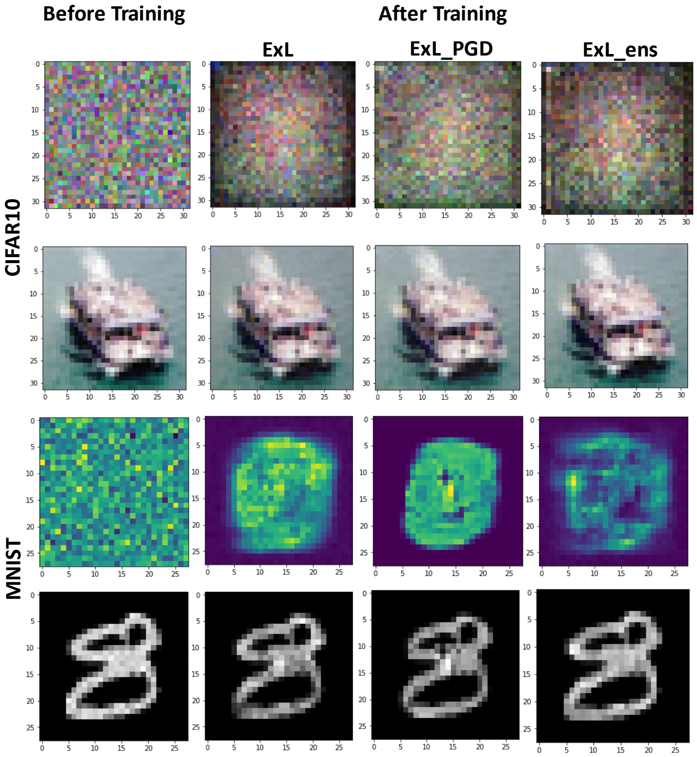


Figure A5: Here, we show the noise templates learnt with noise modeling corresponding to different training scenarios of Table 1, 2 in main paper: ExL (only noise modeling), ExL_PGD (noise modeling with PGDAdv training ExL_{PGD}), ExL_ens (noise modeling with EnsAdv training ExL_{ens}) for MNIST and CIFAR10 data. A sample image ($X \times N$) before and after training with different scenarios is shown. The fact that every training technique yields different noise template shows that noise influences the overall optimization. Column 1 shows the noise template and correponding image ($X \times N$) before training, Coulmns 2-4 show the templates after training. Note, noise shown is the mean across 64 templates.

The Pytorch implementation of ResNet-18 architecture for CIFAR10 and ResNext-29 architecture for CIFAR100 were taken from (Github). For CIFAR10/CIFAR100, we use mini-batch SGD with momentum of 0.9, weight decay 5e-4 and batch size 64 for training the weight parameters of the models. A detailed description of the learning rate and epochs for ResNet18 model (corresponding to Table 2 in main paper) is shown in Table A1. Similarly, Table A2 shows the parameters for ResNext-29 model. The hyperparmeters corresponding to each scenario (of Table A1, A2) are shown in Rows1-6 under *Target* type. The hyperparameters for the source model used to attack the target models for BB scenarios is shown in Row 7/8 under *Source* type. We use BB attacks from the SGD trained source model to attack $SGD, ExL, ExL_{ens}, PGD_{ens}$. We use BB attacks from a model trained with PGD adversarial training ($\epsilon = 8/255$, *step-size=2/255 over 7 steps*) to craft strong BB attacks on SGD_{PGD}, ExL_{PGD}. The model used to generate black box adversaries to augment the training dataset of the SGD_{ens}, ExL_{ens} target models is shown in Row 9 under *EnsAdv* type.

How to conduct Ensemble Adversarial Training? Furthermore, in all our experiments, for EnsAdv training (SGD_{ens}), we use a slightly different approach than Kurakin et al. (2016). Instead of using a weighted loss function that controls the relative weight of adversarial/clean examples in the overall loss computation, we use a different learning rate η_{adv}/η ($\eta_{adv} < \eta$) when training with adversarial/clean inputs, respectively, to learn the network parameters. Accordingly, while performing adversarial training with explainable learning (ExL_{ens}), the noise modeling learning rate in addition to overall learning rate, η_{adv}/η, for adversarial/clean inputs is also different, $\eta_{noise_{adv}}/\eta_{noise}$ ($\eta_{noise_{adv}} < \eta_{noise}$).

How to conduct PGD Adversarial Training? For PGD adversarial training (SGD_{PGD}), we used the techniques suggested in (Kannan et al., 2018). Kannan et al. (2018) propose that training on a mixture of clean and adversarial examples (generated using PGD attack), instead of literally solving the min-max problem described by (Madry et al., 2017) yields better performance. In fact, this helps maintain good accuracy on both clean and adversarial examples. Like EnsAdv training, here as well, we use a different learning rate η_{adv}/η ($\eta_{adv} < \eta$) when training with adversarial/clean inputs, respectively, to learn the network parameters. Accordingly, while performing PGD adversarial training with explainable learning (ExL_{PGD}), the noise modeling learning rate in addition to overall learning rate, η_{adv}/η, for adversarial/clean inputs is also different, $\eta_{noise_{adv}}/\eta_{noise}$ ($\eta_{noise_{adv}} < \eta_{noise}$)

Note, the adversarial inputs for EnsAdv training of a target model are created using BB adversaries generated by R-FGSM from a source (shown in Row 9 of Table A1, A2), while PGDAdv training uses WB adversaries created with PGD attack from the same target model. We also show the test accuracy (on clean data) for each model in Table A1,A2 for reference. Note, the learning rate in each case decays by a factor of 0.1 every 20/30 epochs (Column 5 in Table A1, A2).

Table A1: **Hyperparameter Table for training ResNet18 models on CIFAR10 data**

Model Type	Training Method	Epochs	η/η_{adv}	η, η_{adv} decay/step-size	$\eta_{noise}/\eta_{noise_{adv}}$	$\eta_{noise}, \eta_{noise_{adv}}$ decay/step-size	Test Accuracy in (%)
Target	SGD	120	0.1/–	0.1/30	–	–	88.8
	ExL	120	0.1/–	0.1/30	0.001/–	0.1/30	87.1
	SGD_{ens}	80	0.1/0.05	0.1/30	–	–	86.3
	ExL_{ens}	120	0.1/0.05	0.1/30	0.001/0.0005	0.1/30	86.4
	SGD_{PGD}	122	0.1/0.1	0.1/20	–	–	83.2
	ExL_{PGD}	122	0.1/0.1	0.1/20	0.001/0.0005	0.1/20	83
Source	SGD	300	0.1/–	0.1/100	–	–	89
	$PGDAdv$	122	0.1/0.1	0.1/20	–	–	83
EnsAdv	SGD	31	0.1/–	0.1/30	–	–	81

Table A2: **Hyperparameter Table for training ResNext29 models on CIFAR100 data**

Model Type	Training Method	Epochs	η/η_{adv}	η, η_{adv} decay/step-size	$\eta_{noise}/\eta_{noise_{adv}}$	$\eta_{noise}, \eta_{noise_{adv}}$ decay/step-size	Test Accuracy in (%)
Target	SGD	100	0.1/–	0.1/40	–	–	71
	ExL	58	0.1/–	0.1/20	0.001/–	0.1/20	69.4
	SGD_{ens}	42	0.1/0.05	0.1/20	–	–	69.8
	ExL_{ens}	48	0.1/0.05	0.1/20	0.001/0.0005	0.1/20	67.3
	SGD_{PGD}	52	0.1/0.05	0.1/20	–	–	71.6
	ExL_{PGD}	52	0.1/0.05	0.1/20	0.001/0.0005	0.1/20	69
Source	SGD	34	0.1/–	0.1/10	–	–	67.2
	$PGDAdv$	48	0.1/0.05	0.1/20	–	–	68.4
EnsAdv	SGD	45	0.1/–	0.1/20	–	–	71.3

For MNIST, we use 2 different architectures as source/ target models. ConvNet1: 32C-M-64C-M-1024FC is the model used as target. ConvNet2: 10C-M-20C-M-320FC is the model used as source. Here, we use mini-batch SGD with momentum of 0.5, batch size 64, for training the weight parameters. Table A3 shows the hyperparameters used to train the models in Table 1 of main paper. The notations here are similar to that of Table A1. Note, the source model trained with PGDAdv training to craft BB attacks on ExL_{PGD}, SGD_{PGD} was trained with $\epsilon = 0.3$, *step-size=0.01 over 40 steps*.

C.1 Model Description for Fig. 2 in Main Paper

We use mini-batch SGD with momentum of 0.9, weight decay 5e-4 and batch size 64 for training the weight parameters of the models in Table A4.

Table A3: **Hyperparameter Table for training ConvNet1/ConvNet2 models on MNIST data**

Model Type	Training Method	Epochs	η/η_{adv}	η, η_{adv} decay/step-size	$\eta_{noise}/\eta_{noise_{adv}}$	$\eta_{noise}, \eta_{noise_{adv}}$ decay/step-size	Test Accuracy in (%)
Target ConvNet1	SGD	100	0.01/–	0.1/50	–	–	99.1
	ExL	150	0.01/–	0.1/50	0.001/–	0.1/50	99.2
	SGD_{ens}	64	0.01/0.005	0.1/30	–	–	99
	ExL_{ens}	32	0.01/0.005	0.1/30	0.001/3.3e-5	0.1/30	99.1
	SGD_{PGD}	142	0.01/0.01	0.1/30	–	–	97.9
	ExL_{PGD}	162	0.01/0.01	0.1/30	1e-4/1e-5	0.1/30	98
Source	SGD	15	0.01/–	–/–	–	–	98.6
(ConvNet2)	$PGDAdv$	128	0.01/0.01	0.1/30	–	–	97
EnsAdv ConvNet1	SGD	15	0.01/–	–/–	–	–	98.8

Table A4: **Hyperparameter Table for training ResNet18 models on CIFAR10 data for different types of noise modeling ($X + N, X \times N$) with all/ only negative gradient $\nabla\mathcal{L}_N$**

Noise Modeling Type	Gradient $\nabla\mathcal{L}_N$	Epochs	η	η decay/step-size	η_{noise}	η_{noise} decay/step-size	Test Accuracy in (%)
$X + N$	Negative	120	0.1	0.1/30	0.001	0.1/30	78.1
$X + N$	All	120	0.1	0.1/30	0.001	0.1/30	77.1
$X \times N$	Negative	120	0.1	0.1/30	0.001	0.1/30	87.1
$X \times N$	All	120	0.1	0.1/30	0.001	0.1/30	85.1
SGD	-	120	0.1	0.1/30	-	-	88.9

D Appendix D: Implicit generative modeling of noise acquires adversarial knowledge

Intuitively, we can justify adversarial robustness inherited with noise modeling in two ways: First, by integrating noise during training, we allow a model to explore multiple directions within the vicinity of the data point (thereby incorporating more off-manifold data) and hence inculcate that knowledge in its underlying behavior. Second, we note that noise learnt with ExL inherits the input data characteristics (i.e. $\mathbb{N} \subset X$) and that the noise-modeling direction ($\nabla_N\mathcal{L}$) is aligned with the loss gradient, $\nabla_X\mathcal{L}$ (that is also used to calculate the adversarial inputs, $X_{adv} = X + \epsilon sign(\nabla_X\mathcal{L})$). This ensures that the exploration direction coincides with certain adversarial directions improving the model's generalization capability in such spaces. Note, for fully guaranteed adversarial robustness as per Eqn. 1 in main paper, the joint input/output distribution ($p(X|Y), p(Y)$) has to be realized in addition to the noise modeling and $\mathbb{N}$ should span the entire space of adversarial/off-manifold data.