10-708 Project Final Report: Randomized Deep Learning Models against Adversarial Attacks

Zhuoran Liu, Ninghao Sha[1]

1. Introduction

Modern deep learning architectures have received critical acclaim of their performance in learning a broad spectrum of tasks, such as object recognition, speech processing, text generation, and modeling complex systems. Sitting at the core of this domain is the challenge of deploying robust models for industrial application: different from a research-driven environment where datasets are well-prepared, data acquisition in real world is often vulnerable to adversarial attacks. Unfortunately, it has been shown that machine learning models are typically susceptible to artificially perturbed *adversarial samples* [(Dalvi, 2004)]. Such an example is provided in figure 1.

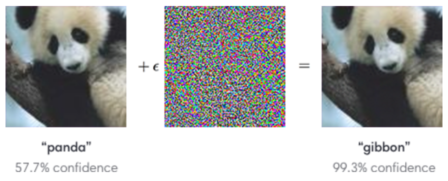

Figure 1. An injection of noise can cause a CNN classifier to mis-classify a panda to gibbon, with no visible change to the original image.

Adversarial examples become particularly problematic in security sensitive domains, such as autonomous driving. In order to develop models that are robust to adversarial examples, it is necessary to know the enemy. Thus, we will first introduce existing approaches to generate adversarial samples, followed by an analysis of current state-of-the-art models for adversarial learning.

2. Literature Review for Adversarial Attacks

Here we assume that the goal of the adversary is to come up with examples that are perceptually indistinguishable from clean inputs but mis-classified by the target model. The most common recipe for a white-box adversarial attack example includes a clean input, the target model in training mode with accessible gradients. After a forward pass, gradients of an adversary-defined objective are back-propagated onto the clean input, revealing the adversarial perturbations that would confuse the model once added to the input. Such white-box gradient-based adversarial examples can be generated either by taking a single step along the gradient (*one-step* methods) or taking steps iteratively until some stopping criterion is met (*iterative* methods). [(Kurakin et al., 2016a)]. On the other hand, we also have black-box adversarial attacks, which requires much less information and therefore is much more accessible in real-world setting. In the black-box setting, the adversaries can easily craft adversarial examples even without any internal knowledge of the target network. In our experiment section, we evaluate the robustness of different model architectures against two white-box attacks, Fast Gradient Sign Attack(FGSM) and Deep Fool Attack, and one black-box attack named LocalSearch. A brief review of each attack mentioned above, and also some related works on adversarial attacks will be discussed below.

Define the loss $L(X, y^{target})$ as a function of input X and its target label y^{target}, which regular, non-adversarial training tries to minimize. The adversary could choose to maximize $L(X, y^{target})$, tweaking the input such that the model is less likely to classify it correctly. The **fast gradient sign method (FGSM)** proposed in [(Goodfellow et al., 2014)] generates adversarial examples by taking a single step:

$$X^{adv} = X + \epsilon sign(\nabla_X L(X, y^{target}))$$

where the size of the perturbation ϵ is often subject to some restrictions [(Kurakin et al., 2016a)]. A common implementation of the FGSM attack is to gradually increase the magnitude of ϵ until the image is misclassified. Its iterative extension named **basic iterative method** is specified in [(Kurakin et al., 2016b)] and has the following update rule:

$$X_0^{adv} = X,$$
$$X_{N+1}^{adv} = Clip_{X,\epsilon}\left\{X_N^{adv} + \alpha sign(\nabla_X L(X_N^{adv}, y^{target}))\right\}$$

where α regulates the size of update on each step and the total size of perturbation is capped at ϵ using $Clip_{X,\epsilon}$ [1].

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

[1]Here we borrow the notation from (Kurakin et al., 2016a).

The DeepFool attack used in our experiment is another iterative attack with also takes into account the L2-norm of the gradient when computing the update rule. The original paper suggests that the proposed method generates adversarial perturbations which are hardly perceptible, while the fast gradient sign method outputs a perturbation image with higher norm. Such pattern is observed on both MNIST and CIFAR-10 dataset using the state-of-the-art architectures.

Algorithm 1 DeepFool Algorithm

```
DeepFool Algorithm(binary case)
Initialize $x_0 \leftarrow x, i \leftarrow 0$
While sign(f(x$_i$)) $= sign(f(x_0))$ :
$r_i \leftarrow \frac{f(x_i)}{||\nabla f(x_i)||_2^2} \nabla f(x_i)$
$x_{i+1} \leftarrow x_i + r_i$
$i \leftarrow i + 1$
```

All methods mentioned above use information about the true label y^{target} while generating adversarial examples from clean ones, making these attacks potentially defective due to the "label leaking" phenomenon discussed in [(Kurakin et al., 2016a)]. The adversary could also choose not to reveal y^{target} by providing a misleading "target" y^{false} in the objective. The **one-step target class method** tweaks the input by a single update such that the model is more likely to output the falsified target:

$$X^{adv} = X - \epsilon sign(\nabla_X L(X, y^{false}))$$

The heuristics of choosing y^{false} varies but [(Kurakin et al., 2016b)] suggested picking y^{false} to be the least likely model output given the clean input. The **iterative least-likely class method** follows that heuristics and is the iterative counterpart of the one-step target class method:

$$X_0^{adv} = X \tag{1}$$

$$X_{N+1}^{adv} = Clip_{X,\epsilon}\left\{X_N^{adv} - \alpha sign(\nabla_X L(X_N^{adv}, y^{false}))\right\} \tag{2}$$

Besides calculating a global perturbation on the entire input, in the context of image classification, [(Papernot et al., 2015a)] also proposed an algorithm that greedily searches for the pixel that would fool the model the most once modified. All of the above-mentioned strategies require access of model gradients to generate adversarial samples. In real life the adversary may or may not have access to gradients of the target model. However, the above methods could be used on a surrogate model, and the adversarial examples produced using the surrogate could frequently trick the target model due to a property called transferability.

$Clip_{X,\epsilon}(A)$ clips A element-wise such that $A_{i,j} \in [X_{i,j} - \epsilon, X_{i,j} + \epsilon]$.

In the case of black-box attack, the adversaries do not have access to model architecture and therefore has no internal knowledge of the target network. These kind of methods treat the network as an oracle and only assumes that the output of the network can be observed on the probed inputs. The LocalSearch attack is accomplished by carefully constructing a small set of pixels to perturb by using the idea of greedy local search. This is an extension to a simple adversarial attack, which is randomly selecting a single pixel and apply a strong perturbation to it in order to misclassify the input image. The LocalSearch attack is also an iterative procedure, where in each round a local neighborhood is used to refine the current image and in process minimizing the probability of the network assigning high confidence scores to the true class label. This approach identifies pixels with high saliency scores but without explicitly using any gradient information.

3. Methods for Defensing Against Adversaries

On the defenders' side, proposed measures against adversarial attacks include input validation and preprocessing, adversarial training, defensive distillation and architecture modifications. We review some existing methods and discuss how randomized training/models such as stochastic delta rule could increase model robustness against adversaries.

Adversarial Training

Adversarial training increases model robustness by feeding in adversarial examples to the model during training. The standard practice is to generate adversarial examples from a subset of the incoming batch of clean inputs on the fly. The model is then trained on the mixed batch of clean and adversarial inputs. However in order to do that, a specific method for generating adversarial examples must be assumed, preventing adversarial training from being adaptive to different attack methods. For example, (Kurakin et al., 2016a) showed that models adversarially trained using *one-step* methods are fooled easily by adversarial examples generated using *iterative* methods; models adversarially trained using a fixed ϵ could even fail to generalize to adversarial examples created using different ϵ values.

Defensive Distillation

Distillation was originally proposed in the context for model compression, aiming to transfer learned knowledge from larger, more complex models to more compact and computationally efficient models [(Hinton et al., 2015)]. *Defensive distillation* was first proposed by [(Papernot et al., 2015b)] as a training regime to increase model robustness against adversaries. The goal of *defensive distillation* is not transfer learning as how distillation was originally proposed, but rather to train models to have smoother gradient surfaces with regard to the input – such that small steps in the input space do not change model output significantly. While smoothing out the gradients that adversaries usually use to create adversarial examples is effective in the setting shown by the original paper, [(Carlini & Wagner, 2016)] pointed out that Papernot's attack assumed by [(Papernot et al., 2015b)] could miss out potentially strong attacks and that models trained with *defensive distillation* possess no advantage against a modified version of Papernot's attack when compared to regularly trained models. Works such as [(Belagiannis, 2018)] investigates the effect of network compression solely for the purpose of transferring model knowledge, but discovers the effect of robustness against adversaries as a side product.

Randomized Methods & Models

Randomized training methods seek to improve the robustness of deep models by introducing randomness, irrespective of benign or adversarial samples, during the training process. For example, [(Xie)] introduces a random resizing layer and/or zero-padding layer prior to the regular architectures of CNNs. Through extensive experimental evaluation, the authors discovered that this method is particularly effective against iterative attacks, while other methods introduced above are better at handling single-step attacks. A combination of both methods, as the authors argue, achieve best performance against arbitrary adversaries.

4. Baseline Randomized Model: DropConnect

DropConnect is a generalization of Dropout layer for regularizing large fully-connected layer within neural network. In the case of Dropout layer, a random set of outputs from the activation layer are muted in order to force the network to learn more robust features. A random subsets of the weights within the fully-connected layers are set to zero. The input features are then multiplied by this randomized weight matrix before passing through the activation. Figure 2 and 3-5 are the visualization of the DropConnect architecture, and a comparison among the regular fully-connected network, a Dropout network and a DropConnect network.

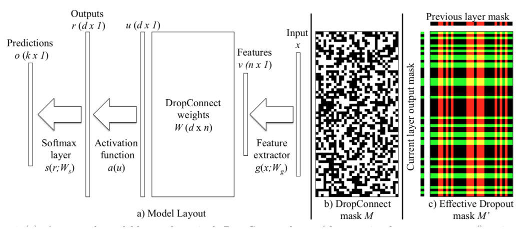
Predictions o (k x 1)
Outputs r (d x 1)
u (d x 1)
Features v (n x 1)
Input x
Previous layer mask
DropConnect weights W (d x n)
Softmax layer $s(r;W_s)$
Activation function $a(u)$
Feature extractor $g(x;W_g)$
Current layer output mask
a) Model Layout
b) DropConnect mask M
c) Effective Dropout mask M'

Figure 2. DropConnect Architecture

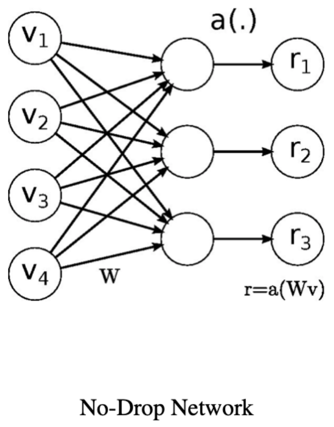

Figure 3. Regular full-connected network

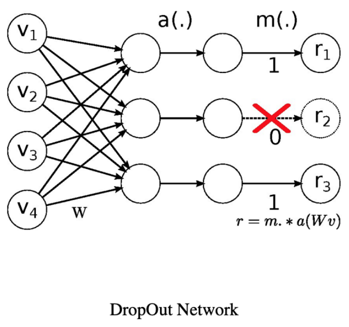

Figure 4. Dropout network

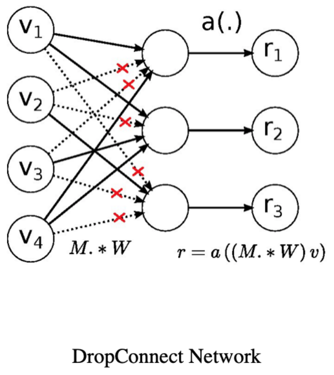

Figure 5. DropConnect network

5. The Stochastic Delta Rule

In this section, we provide the specifics of our proposed SDR training routine, along with its fine-grained variant. Along this trajectory, we will raise the issue of practical implementation concerns, the connection between SDR and regularization, and a qualitative explanation of feasibility of SDR-augmented training routine in improving model generalizability and robustness against adversaries.

First introduce in [(**?**)], SDR is revisited under the deep learning setting in [(Frazier-Logue & Hanson., 2018)]. During the forward pass of a SDR-equipped model, a parameter w_j is not regarded as fixed values, but are rather random variables sampled from an arbitrary distribution specified by parameters θ_j. The choice of such distribution is arbitrary. For the purpose of our experiments, we assume that model parameters follow a normal distribution, and are independent with each other. At training iteration t, a model with the SDR training routine samples model parameters $w^{(t)} \sim N(\mu^{(t)}, \Sigma^{(t)})$, fit the current batch with respect to the sampled parameters, and perform the following updates:

$$\begin{aligned} \mu^{(t+1)} &\leftarrow \mu^{(t)} - \alpha \nabla_w L(X, y, w^{(t)}) \\ \Sigma^{(t+1)} &\leftarrow \Sigma^{(t)} + \beta |\nabla_w L(X, y, w^{(t)})| \end{aligned} \tag{3}$$

where α, β are step sizes for the mean and variance respectively. On one hand, it can be readily observe that with $\beta = 0$ and zero-intialization of the covariance matrix, only the mean parameters are updated, and we recover the training routine without SDR. On the other hand, [(Frazier-Logue & Hanson., 2018)] states that sampling parameters from a binomial distribution with mean Dp and variance $Dp(1-p)$, SDR could be regarded as a special case of dropout with probability p, only in this case the variance is not updated with respect to information gathered from the gradients.

5.1. SDR Update with Scheduled Variance Decay

The first SDR training routine we consider is termed `SDR-Decay`, formally presented in algorithm 5.1. In this algorithm, aside from controlling parameter variances by end-to-end updates with $\nabla_w \partial L(X, y, w)$, we further anneal the variance by ζ to ensure asymptotically decaying variances. As training progresses, we shrink the variance so as to sample progressively concentrated parameters around the mean. We further introduce the decay schedule, τ, to avoid over shrinkage and allow sufficient exploration of our model within the parameter space. At test time, we compute the forward pass of our model directly with parameter means.

Algorithm 2 `SDR-Decay`

```
input dataset $\{(X_i, y_i)\}_{i=1}^n$, decay schedule $\tau$, decay rate $\zeta \in (0,1]$, batch size $B$
Initialize model parameters $\mu^{(0,0)}, \Sigma^{(0,0)}$
num-batches ← $n // B$
for $e = 1, 2, \cdots$ until convergence:
for $b = 1, 2, \cdots,$ num-batches
Sample batch parameter weights $w^{(b,e)} \sim N(\mu^{(b,e)}, \Sigma^{(b,e)})$
Compute forward pass of network with respect to $w^{(b,e)}$
Perform SDR parameter updates with respect to equations 3
If $b \% \tau = 0$: $\Sigma^{(b,e)} \leftarrow \zeta \Sigma^{(b,e)}$
End for
End for
```

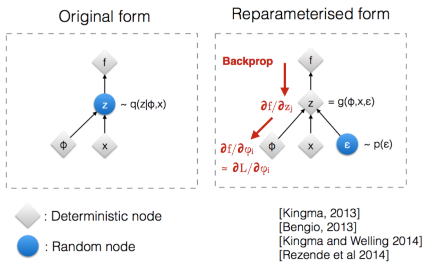

Figure 6. Reparameterization trick of `SDR-Decay` update

5.2. Reparametrization Trick

With this formulation of the update rule, we require the loss function to be differentiable with respect to the parameters μ and Σ. However, the loss value is computed based on realizations of many random variables along the forward pass, which are not explicitly differentaible with respect to their parameters. In order to combat this issue, we use the reparametrization trick, as illustrated in figure 6, to make the network capable of backpropagating through random nodes. This technique essentially transfers the undeterministic nature of the weight to another source of randomness, which then allows the randomly generated weights to be differentiable with respect to its parameters. This trick is most known for its application in Variational Autoencoder(VAE). Essentially we want to sample each weight from its posterior distribution, say $q(z|\phi, x)$, where x denotes the data and ϕ denote the parameters in the distribution, which need to be updated during the training process. The problem at hand is that we want to compute

$$\nabla_\phi \mathbb{E}_{z \sim q(z|\phi,x)}[f(x,z)]$$

. This is similar to a score function estimator used in reinforcement learning, which has a high variance and very often leads to difficulties in model learning. In order to compute this term, we typically utilize estimation method like Monte-Carlo to derive an estimate, instead of direct back-propagation.

In order to alleviate this learning issue, the reparametrization trick is often used, which allows us to estimate $q_\phi(z|x)$ as a two-stage process. First, we sample a noise variable ϵ from some common distribution(usually the standard normal). Then we create a deterministic transformation $g_\phi(z|x)$ which allows us to map the generated random noise to a more complex distribution(for example a generalized normal distribution). For most commonly seen distributions, such mapping indeed can be found. For example, suppose the posterior distribution of a weight follows the following distribution:

$$z \sim q_\phi(z|x) \sim \mathbb{N}(\mu, \sigma)$$

. Now instead of directly samping from $q_\phi(z|x)$, we generate the random variable by applying a transformation on ϵ:

$$z \sim \mu + \epsilon * \sigma$$

. With this transformation, we can rewrite the gradient term we need to compute in a more estimation-friendly form:

$$\nabla_\phi \mathbb{E}_{z\sim q(z|\phi,x)}[f(x,z)] \tag{4}$$
$$= \mathbb{E}_{\epsilon\sim p(\epsilon)}[\nabla_\phi f(x, g_\phi(\epsilon, x))] \tag{5}$$

An important observation is that now the expectation is with respect to the distribution of the random noise, and is independent of the gradient parameter ϕ. Therefore we can put the gradient directly inside the expectation. The variance of this new form of gradient estimation is much lower, which greatly improves the efficiency of the training process.

5.3. SDR Update with Learnable Variance

Algorithm 5.1 updates parameter means and variances with $\nabla_w \partial L(X, y, w)$, requiring backward pass on only one set of parameters w for each batch. We can, however, let both means and variances be fully learnable and have them updated with $\nabla_w \partial L(X, y, w)$ and $\nabla_\Sigma \partial L(X, y, w)$ respectively. The resulting algorithm, termed `SDR-Learnable`, produces a more realistic learning paradigm that approximates the behavior of variable parameters with higher fidelity. The drawback of such approach, of course, is doubling the computation required to compute the gradients. At test time, we sample parameters from learned means and variances, and perform inference on input data with sampled parameters.

5.3.1. SDR-Learnable in Adversarial Learning

The inference procedure of `SDR-Learnable` produces variable predictions with same inputs, which motivates us to investigate its robustness against adversarial samples. Our qualitative motivation for this hypothesis is as follows: adversarial attacks are designed to lead models to mis-classify, while inducing no recognizable changes to inputs subject to human examination. For example, an adversarial image of a digit in the MNIST dataset should be very close numerically to its original counterpart, as suggested by figure 7. Thus, the decision surface of a model should not only cater to singular data points provided in training data, but also a vicinity region in the sample space sharing similar topologies.

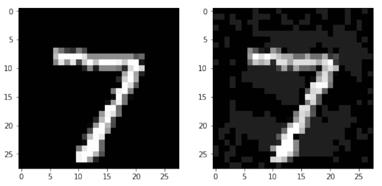

Figure 7. Left: an image of digit 7 from the MNIST dataset; right: an adversarial image of the said image, generated with FGSM attack on a trained MLP for this project

The variable treatment of parameters in `SDR-Learnable` is a step towards defensive strategies in two aspects: (1). the most effective gradient attack direction is computed with respect to one sampled parameter instance, and is less effective for another; (2). instead of fitting data at singular points, models with variable parameters create a decision region subject to parameter distribution, hence allowing more robust prediction against adversarial samples.

5.3.2. SDR-Learnable in Generalization

The description of decision regions, rather than spiky predictions, of `SDR-Learnable` naturally lead us to investigate its relationship to model generalization. In particular, we are interested in whether the inclusion of variable model parameters improves out-of-sample performance, and how do variances behave as training progresses. We present the following theorem as a first step towards the analysis.

Theorem 1. *Let $X \in \mathbb{R}^{N\times D}$ and $y \in \mathbb{R}^N$ be fixed, and $w \in \mathbb{R}^D$ be a random vector with $\mathbb{E}[w] = \mu$, $\text{Cov}[w] = \Sigma$, then the risk of a linear regression model $\hat{y} = Xw$ takes the form*

$$\mathbb{E}_w[||y - Xw||^2] = ||y - X\mu||^2 + ||X\Sigma^{1/2}||^2$$

Proof. With simple algebra of expectation, we observe that

$$\mathbb{E}_w[||y - Xw||^2] = ||y||^2 - 2y^T X\mathbb{E}_w[w] + \sum_{i=1}^N \mathbb{E}_w[(X_i^T w)^2]$$
$$= ||y||^2 - 2y^T X\mathbb{E}_w[w] + \sum_{i=1}^N \left(\text{Var}[X_i^T w] + (X_i^T \mathbb{E}_w[w])^2\right)$$
$$= ||y||^2 - 2y^T X\mu + \sum_{i=1}^N (X_i^2\mu)^2 + \sum_{i=1}^N X_i^T \Sigma X_i$$
$$= ||y - Xw||^2 + ||X\Sigma^{1/2}||^2.$$

□

It can be readily observe that under the stylized linear regression model, `SDR-Learnable` is equivalent to regularizing parameter variances Σ with penalty matrix X. In neural network training, we expect that the the $\mathcal{L}$-2 norm of Σ to decay progressively, hence leading to more concentrated parameter samples. Notice that the decay step $\Sigma \leftarrow \zeta\Sigma$ in `SDR-Decay` is a step towards artificial control of the magnitude of parameter variances, mimicing the behavior of `SDR-Learnable`.

However, it is noteworthy that the parameter distribution does not necessarily lead to better generalization. Thus, aside from performing inference on one set of sampled parameters at test time, we inference for arbitrary number of times, and take the majority vote of these predictions.

6. Experiments & Results

So far, we have explored qualitative and theoretical aspects of SDR-based routine for learning deep neural networks. In the experiment section, we will further evaluate the efficacy of SDR in model generalization and robustness against adversarial samples. We will also verify the regularization effect of parameter variances in `SDR-Learnable`, and explore the relationship between prediction accuracy and adversarial robustness. Implementation of baselines and SDR related models can be found on our github repository: `https://github.com/ssandyshaa/SDR`.

6.1. SDR in Model Generalization

We evaluate the convergence property of `SDR-Decay` and `SDR-Learnable` on three datasets: MNIST, Sequential MNIST, and CIFAR-10, with MLP, LSTM, and ResNet-18 architectures respectively. All of our subsequent experiments on conducted on these datasets and models, with details provided in the table 1.

We train all models for 20 epochs with batch size of 64 on a GeForce RTX 2080 GPU, using the Adam optimizer with a learning rate of 0.001. For the `SDR-Decay` learning routine, we let $\beta = 0.05$ and $\zeta = 0.7$, with parameter variance decay applied twice per epoch. Due to time constraint, we only conducted experiments of `SDR-Learnable` on MLP and ResNet, with majority votes taken from 50 ensembles per input. The plots of validation accuracies are shown in figure 8 and best performance shown in table 1.

As we can readily observe, `SDR-Decay` outperforms vanilla method in terms of out-of-sample accuracy and convergence behavior for ResNet and LSTM. We have reasons to believe that the parameter distribution in the decay method is encouraging our model to explore different optimums, thus leading to better overall performance.

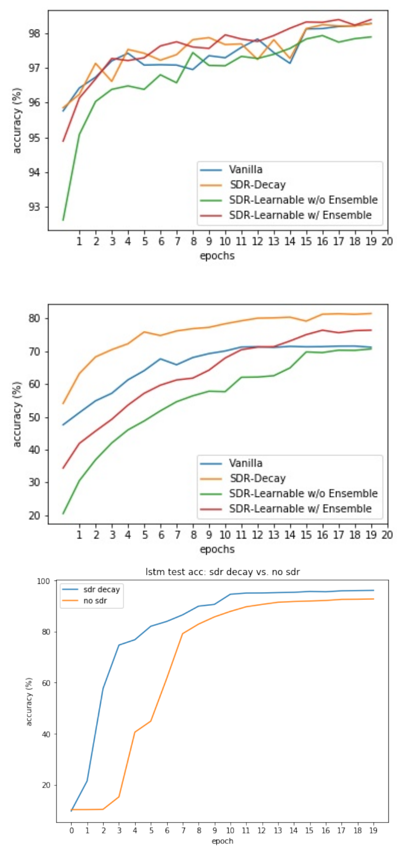

Figure 8. Validation accuracies for: MLP for MNIST (left), ResNet-18 for CIFAR-10 (middle), LSTM for Sequential MNIST (right)

`SDR-Learnable` with ensemble tops the MLP digit classification task. In this case, we observe that learnable methods is capable of producing smoother accuracy curves than vanilla and decay methods, which suffer from a performance drop around 15 epochs. Nonetheless, `SDR-Learnable` suffer from a significant performance drop while learning with ResNet-18, which could be attributed to the doubling of learnable parameters in this setting ($\sim$ 11m $\rightarrow \sim$ 22m). In this case, 20 epochs of training may not suffice to guarantee convergence behavior of parameter variances. However, through taking majority vote of ensemble networks, we recover true parameters mean with Law of Large Numbers, and thus remedy the ensemble method to beat the vanilla method by a significant margin.

Overall, we learn that `SDR-Decay` is capable of significantly improving model performance with no additional GPU computation, with `SDR-Learnable` performing on par in smaller networks, such as a MLP. The latter suffer from a performance drop in larger networks, which can be remedied by taking majority vote of ensemble predictions.

Table 1. Model specification for experiments

Model	Architecture	Activation	Vanilla	Decay	Learnable	Ensemble
MLP	784; 100; 100; 100	ReLU	98.28	98.27	97.93	**98.39**
ResNet-18	As in [7]	ReLU	71.6	**81.53**	70.72	76.43
LSTM	784; 130	ReLU	87.8	**91.3**	89.7	90.6

6.2. SDR in Adversarial Learning

We conducted our experiments on the MNIST dataset. We tested 4 models: vanilla 3-layer MLP, 3-layer `SDR-decay` MLP, 3-layer `SDR-Learnable` MLP and a 3-layer Drop-Connect MLP with a drop probability of 0.2. Notice that in Algorithm 2, we shrink the parameter variance by ζ in order to mimic the regularizing behavior demonstrated in Theorem 1. The model specifics are listed in Table 2.

We trained all four models using regular samples of MNIST and obtained validation prediction accuracy. The `SDR-Learnable` model achieves a slightly lower accuracy($\sim 2\%$) than the other three models, which is not surprising given the additional source of randomness in the inference phase. However, its prediction accuracy on adversarial samples is far more superior than all the other models. On these adversarial samples, the `SDR-Learnable` MLP achieves almost the same prediction accuracy as on the uncontaminated dataset, while the other models are very vulnerable to such attack. The results are summarized in the following table:

Despite previous experiments being conducted only on fully connected layers, Stochastic Delta Rule is easily generalized to other layer architecture, such as convolution layers. We thus modified the convolution layer in the Resnet18 model and trained it on the CIFAR dataset. However the accuracy we obtained is about 10 percent lower than the regular Resnet18. Therefore we need to figure out a heuristic to control the randomness in the inference phase before testing model robustness against adversaries.

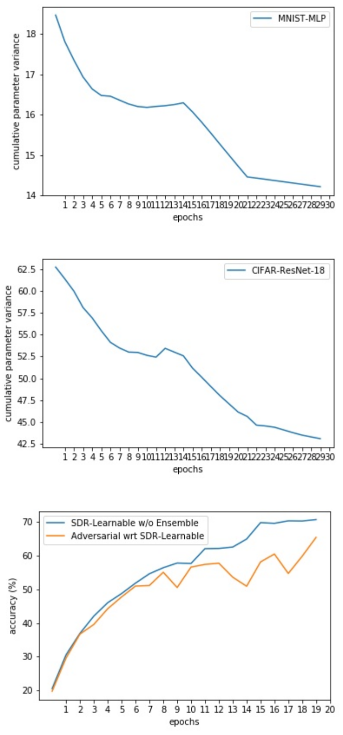

Figure 9. Cumulative sum of parameter variances for MLP (first) and ResNet-18 (middle), and classification accuracy on validation set and FGSM adversarial samples for CIFAR (last)

6.3. Variance Shrinkage Effect of `SDR-Learnable`

Theorem 1 states that as training progresses, parameter variances shrinks asymptotically, leading to more concentrated parameter instances. However, our previous experiments indicate that models with lower uncertainty (in terms of parameter variance) leads to more accurate prediction, but are more sensitive to adversarial attacks. This behavior suggests a trade-off between these two objectives. In this section, we verify empirically that variance decay holds, followed by an investigation of classification accuracy vs. adversarial accuracy along training.

Figure 9 verifies our hypothesis. Specifically, we observe that the cumulative parameter variance is progressively decreasing along training, despite both encountering saddle points at around 15 epochs. The magnitude of variance for CIFAR is also much larger than that of MNIST (40-60 vs. 14-18), introducing a greater degree of variability in making predictions.

To further investigate this trade-off, we apply FGSM method on the trained ResNet-18 model after each epoch, and generate 20 sets of adversarial images, each containing the same 1000 samples from the CIFAR-10 validation set. We then evaluate classification accuracies on the adversarial samples against validation accuracies, as plotted in the right subplot in figure 9. As training progresses, we observe a widening gap between the validation curve and the adversarial curve, which confirms our hypothesis that sharper distribution leads to higher out-of-sample performance at the cost of robustness against adversarial examples.

	Vanilla MLP	SDR-decay MLP	Learnable SDR MLP	DropConnect
Regular Samples	97.59%	97.55%	95.55%	97.95%
FGSM Attack	0%	4.61%	**94.86%**	45.9%
DeepFool Attack	0%	3.87%	40.15%	**45.48%**
LocalSearch Attack	0%	2.85%	40.15%	26.53%

Table 2. Model performance without and under adversarial attack.

7. Conclusion

In this project, we revisited the classical stochastic delta rule (SDR) and performed an evaluative study on its performance in out-of-sample prediction and robustness against adversarial samples. Our results have shown that `SDR-Decay`, an update rule which performs both parameter updates with gradients with respect to mean, leads to superior performance in terms of convergence and classification accuracy with minimal extra computation cost. On the other hand, the fully learnable paradigm `SDR-Learnable` demonstrates extraordinary robustness against adversarial samples, at the cost of lower prediction accuracy. To remedy this loss, we propose an ensemble method which predicts based on the majority vote of predicted instances, which yields higher prediction accuracy but lower adversarial accuracy. This last observation leads us to investigate empirically the trade-off between out-of-sample performance and adversarial robustness.

References

Belagiannis, Vasileios, e. a. Adversarial network compression. *arXiv:1803.10750*, 2018.

Carlini, N. and Wagner, D. A. Defensive distillation is not robust to adversarial examples. *CoRR*, abs/1607.04311, 2016. URL http://arxiv.org/abs/1607.04311.

Dalvi, Nilesh, e. a. Adversarial classification. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004.

Frazier-Logue, N. and Hanson., S. J. Dropout is a special case of the stochastic delta rule: faster and more accurate deep learning. *arXiv:1808.03578*, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv e-prints*, art. arXiv:1412.6572, Dec 2014.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial machine learning at scale. *CoRR*, abs/1611.01236, 2016a. URL http://arxiv.org/abs/1611.01236.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016b. URL http://arxiv.org/abs/1607.02533.

Papernot, N., McDaniel, P. D., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. *CoRR*, abs/1511.07528, 2015a. URL http://arxiv.org/abs/1511.07528.

Papernot, N., McDaniel, P. D., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. *CoRR*, abs/1511.04508, 2015b. URL http://arxiv.org/abs/1511.04508.

Xie, Cihang, e. a. Mitigating adversarial effects through randomization. *arXiv:1711.01991*.