# Analyzing the implicit adversarial robustness of networks trained using SGD

Anirudh Choudhary (ac67@illinois.edu)

17 December 2020

**Abstract**

Stochastic Gradient Descent is the current workhorse of training neural networks and the hyperparameters such as learning rate, batch size and momentum play an important role in the generalization capabilities of the trained networks. It has been shown that certain parameter combinations such as large batch training leads to networks which do not generalize well and have poor adversarial robustness. Moreover, various improvement to SGD's formulation have been proposed to improve convergence and generalization via sampling, noisy gradients or variance-reduction. In this study, we empirically analyze the impact of various hyperparameter configurations on the adversarial robustness of networks trained using vanilla minibatch SGD and compare the performance of minibatch SGD with improved variants of SGD which guarantee better convergence. We aim to understand the implicit robustness of networks trained using SGD and relate it to classification accuracy and generalization capability.

## 1 Background

Deep neural networks(DNN) have achieved impression performance in several domains and are being increasingly adopted in safety-critical tasks such as self-driving cars, fraud detection, malware detection etc. Recently, many studies [6, 5] have shown that DNNs employed in real-world systems are vulnerable to adversarial examples, thus raising safety concerns. Adversarial examples, discovered by Szegedy et al. [22], are inputs intentionally designed by an attacker to force a DNN to generate erroneous outputs. Robustness to adversarial attacks is an important factor to be considered while designing classifiers using DNNs. To defend against adversarial attacks, one of the most popular approach utilized by existing defense methods is augmenting the training dataset with adversarial examples and making the network robust by training it on adversarially perturbed samples explicitly, also known as adversarial training. This procedure is equivalent to an explicit regularization. However, obtaining a naturally robust model without adversarial training is a desirable property. This is a form of implicit regularization and the robustness of the network depends on the loss optimization procedure i.e, networks optimized well with weights converging to flatter stable minimas and smoother class boundaries, have been shown to generalize well during test time. These networks can be learnt using standard training methods such as Stochastic Gradient Descent (SGD) and can attain near-perfect accuracy on certain popular image classification datasets such as CIFAR and ImageNet.

SGD and its variants are being extensively leveraged for training neural networks. Hyperparameters like learning rate, batch size and momentum play an important role in SGD reaching an optimum minima, which generalizes well. Many studies have focused on understandong the impact of hyperparameter tuning on the final accuracy and the network's generalization ability. For instance, Smith et al. [20] and Hoffer et al. [9] highlighted clear rules and relation between the hyperparameters. Goyal et al. [8] showed that Imagenet can be trained quickly by relating learning rate and batch size in a distributed setting. Keskar et al. [14] showed that large batch training leads to sharp minima and suboptimal generalization. They propose a solution to handle the sharp minimas and improving SGD's convergence by reducing the variance in gradient updates. Recent work by Jastrzebski et al. [10] has showed that maintaining a constant ratio between learning rate and batch size leads gradient descent algorithm to converge to flatter minima with better generalization. Moreover, works by [3] have shown that SGD induces a crucial implicit regularization, which prevents over-parameterized neural networks from converging to minima that cannot generalize well. This regularization has been due to an unbiased noise inserted at every iteration, called the gradient noise which has a nontrivial covariance structure [23]. Another line of research [13] has focused on proposing improved variants of SGD with lower variance during gradient updates leading to higher accuracy and faster convergence. These studies employ variance reduction techniques such as importance sampling, control-variates or minibatching to improve training and enable linear convergence of SGD to better minima [4].

However, there has been lesser attention towards exploring the impact of hyperparameter tuning and variations of SGD in terms of guaranteeing simultaneous high accuracy and adversarial robustness. Motivated by the recent works of [17, 24], our study aims to understand the impact of different formulations of SGD as well as hyperparameter settings on the robustness of resulting network against standard adversarial attacks (FGSM/PGD). This is important since as highlighted by Ye et al. [25], adversarial robustness requires a significant larger architectural capacity of the network than that for the natural training with the original training data. This might be a bottleneck for security-critical scenarios which require faster inference and might have resource constrained application systems. Having a better understanding of of how tuning the optimization process can influence the adversarial robustness of the trained network, can enable us to extract the best possible performance given model constraints and with limited dependence on artifically generated data.

Our work could be viewed as understanding the influence of hyperparameters of SGD and the improved variants of SGD proposed recently, on the natural adversarial robustness of networks trained with original clean training data. Previous studies address this issue at the architecture level, without adversarial training or utilize SGD with weight decay to obtain network with better generalization. Specifically, we consider different configurations of learning rate, batch size, momentum and weight decay for SGD, and evaluate the impact of hyperparameter tuning on the adversarial robustness of trained network. In addition, we consider four popular variants of SGD which focus on improving the convergence rate and training of SGD via minibatching (minibatch SGD), variance reduction (Stochastic Variance Reduction Gradient) [2], noisy gradients (minibatch multiplicative SGD) [23] or sampling informative training examples (Selective Backpropagation based SGD) [11]. We compare their classification performance on MNIST and CIFAR-10 datasets. To perform adversarial atatcks, we adversarially peturb the test samples using Fast Gradient Sign Method (FGSM) [7] which adds adversarial noise using gradient updates and Projected Gradient Descent (PGD) [19], which draws adversarial perturbations from an $l_\infty$ ball around input image $x$ and perturbs each pixel value by a quantity within [-$\epsilon$, $\epsilon$].

## 2 Methods

First, we will introduce some notations. Let the training data be given by $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$, where $x$ belongs to input space $\mathcal{X}$ and $y$ belongs to output space $\mathcal{Y}$. We focus on multi-class classification problem in which our aim is to determine a prediction function $f : \mathcal{X} \to \mathcal{Y}$, such that $f(x)$ estimates $y$ accurately. The function $f$ is parameterized by a real vector $\theta \in \mathbb{R}^d$ and here, we $f$ represents a family of neural networks. To determine the parameter $\theta$, we wish to minimize the empirical risk i.e., the expected loss $L(\theta) = \frac{1}{n}\sum_{i=1}^n l(x_i, \theta)$ where $l(x, \theta)$ is the loss incurred with respect to $i^{th}$ sample. Stochastic optimization leverages the stochastic gradient $\hat{g}_i(\theta) = \nabla_\theta l(x_i, \theta)$ to determine the descent direction and minimize the loss over training data.

### 2.1 Vanilla SGD

In its simplest form, vanilla SGD proceeds over iterations, selecting a single sample and updating the parameter $\theta$ by substracting the gradient of loss multiplied by a step-size $\eta$. The pseudocode is given by algorithm:

**Algorithm 1** Vanilla Stochastic Gradient Descent

```
Result:
Initialize parameter $\theta_0$; choose learning rate $\eta > 0$, batch size $b$ and number of iterations $n$
for i=1,2....n do
    Randomly sample an example $x_i$ and compute the loss $l(x_i, \theta_k)$
    Compute gradient of the loss $\hat{g}(\theta_k) = \nabla l(x_i, \theta_k)$
    Update $\theta_{k+1} = \theta_k - \eta \hat{g}(\theta_k)$
end
Output: $\theta_k$
```

In expectation, it can be shown that the sequence $\{\theta_k\}_{k=1}^n$ would converge towards the minimizer of $L$. While vanilla-SGD leads to a fine-grained iterative process, it introduces considerable stochastic noise, which does not allow the algorithm to converge and halts progress. An extreme alternative is using a batch approach, which needs to look at complete training data to perform the parameter update $\theta_{t+1} = \theta_t - \frac{\eta}{n}\sum_{i=1}^{n} g_i(\theta_t)$, thus making it computationally impractical. A popular strategy to overcome variance in vanilla SGD is minibatching i.e., utilizing a random set of examples in the estimate of the gradient, instead of a single example.

## 2.2 Variants of SGD

### 2.2.1 Mini-batch SGD

In minibatch SGD, at each step a subset of training examples known as minibatch $B_t = \{x_1, ..., x_b\}$ comprising $b$ randomly sampled examples (without replacement) is created from $\mathcal{D}$. During each epoch, the full training data is traversed using mini-batches and the parameter update is given by:

$$\theta_{t+1} = \theta_t - \eta\tilde{g} \text{ where } \tilde{g}(\theta_t) = \frac{1}{b}\sum_{i \in B_t}^{b} \hat{g}_i(\theta_t) \tag{1}$$

Batch-size $b$ and learning rate $\eta$ are important hyperparameters for successful training and convergence of SGD. Moreover, SGD typically oscillates across the slopes of ravines i.e., areas where the loss surface curves steeply in one dimension than other, typically observed around local optima. Momentum is used to help accelerate SGD in the correct direction and suppress the oscillations. A lot of research has been performed in evaluating the impact of batch size, learning rate and momentum over SGD's convergence to sharper or flatter minimas and its generalization ability. Here, we highlight few key hyperparameter optimization approaches for SGD and evaluate their impact on the adversarial robustness of trained neural networks $f$.

**Batch size vs learning rate** Larger batch size leads to faster training, however, it has been shown to lead to poor generalization, as shown by [9]. They highlight that training for longer time might improve network's generalization. Recently, [10] have shown that training SGD with maintaining a constant learning rate to batch size ratio helps SGD in converging to a flatter minima. [24] highlight that the adversarial robustness of neural network is correlated to the top eigenvalue of the Hessian and characterize the vulnerability of networks trained with large batch size and varying learning rate to batch size ratio. Tuning the learning rate has been shown to lead to better convergence of SGD to minima that generalizes well. A common approach in tuning learning rate is to decrease (anneal) it using a schedule. Large initial learning rate have been shown to achieve better generalization soon after it is annealed [16]. This is because with initial small learning rate, the network memorizes easy-to-generalize, hard-to-fit patterns. Hence, we also consider evaluating a simple annealing regime for learning rate based on step schedule, wherein the learning rate is reduced by a factor $\gamma$ after certain training epochs, when the training accuracy stabilizes and increases by very smal aamount.

**Momentum** : Momentum helps accelerate SGD in the relevant direction and dampens oscillations by adding a fraction gradient update the previous time step to the current update:

$$\theta_{t+1} = \theta_t - v_t \text{ where } v_t = v_{t-1} + \eta\tilde{g}(\theta_t) \tag{2}$$

Momentum increases updates for dimensions whose gradients point in the same direction and reduces updates for dimensions whose gradient change directions. Despite various empirical studies, there has been lack of clear understanding as to how momentum affects the generalization and convergence. Adding momentum helps in faster convergence and hence, its its impact on generalization and adversarial robustness needs to be analyzed. Sutskever et al. [21] found momentum to be critical to obtain good performance but Ma and Yarats [18] highlighted that momentum may have little practical effect, with varying parameters.

**Regularization** We focus on $L_2$ network parameter regularization, also known as weight decay. Regularization is used since neural networks with rich solution space can easily overfit on training data, which impacts generalization. $L2$ regularization optimizes the loss $L = l(x_i, \theta) + w||\theta||^2$, where w is the rate of weight-decay. Liu et al. [17] showed that adding regularization and momentum allows SGD to overcome random adversarial initialization during training and converge to stable good global minima. They highlight that regularization makes adversarial initialisation a far-worse model and incentivizes SGD to move away from it.

### 2.2.2 Variance-reduced SGD

Minibatch stochastic gradient descent suffers from noisy-gradient estimates. The optimization performance of minibatch SGD near the local minima is limited by the noise due to minibatch sampling. Due to the sampling noise, SGD can only converge using decaying step sizes with sub-linear convergence rate. Various variance reduction methods have been proposed to alleviate the noisy gradient problem by using gradient aggregation which leads to unbiased and low-variance gradient estimates. These methods achieve linear convergence rate with a fixed step size, but at a higher computational cost. Here we focus on Stochastic Variance Reduced Gradient(SVRG) [12]. SVRG comprises of inner and outer loops, wherein, in the outer loop, a large batch gradient $\tilde{g} = \frac{1}{N}\sum_{i=1}^{N} \nabla_{\theta_{mT}} L_i$ is computed and parameter $\theta_{mT}$ is stored, where $N \geqslant b$, and $m$ is the outer loop index. In the inner loop, there reference value of gradient is set to $\theta_{mT}$ and updated using following parameter update:

$$\theta_{mT+t+1} = \theta_{mT} - \eta[\tilde{g}(\theta_{mT+t}) - \tilde{g}(\theta_{mT}) + \bar{g}] \tag{3}$$

where $\tilde{g}(\theta_{mT+t})$ is the current gradient of mini-batch and $\tilde{g}(\theta_{mT})$ is the gradient for stored parameters. The variance is reduced due to stored $\theta_{mT}$ computed over the entire dataset.

### 2.2.3 Noisy SGD

The implicit gradient noise of SGD has been shown to lead to better generalization [3]. The SGD noise has been shown to have a nontrivial covariance structure instead of white noise and acts as an effective regularizer for gradient descent. Due to the higher gradient noise in small-batch SGD, it has been shown to lead to better generalization than large-batch SGD. During each iteration of mini-batch SGD, due to the randomness of mini-batch sampling procedure, the stochasticity of gradient could be characterized as:

$$\tilde{g}(\theta_t) = \nabla_{\theta_t} L(\theta_t) \cdot \mathcal{W}_{sgd} \tag{4}$$

where $\mathcal{W}_{sgd}$ is a random sampling vector representing the mini-batch sampling process. The stochastic gradient is an unbiased estimator of the full gradient $\nabla_{\theta_t}\mathcal{L}(\theta_t)$ since $\mathbb{E}[\mathcal{W}_{sgd}] = \frac{1}{n}1$. The stochastic gradient can be decomposed as :

$$\tilde{g}(\theta_t) = \nabla_{\theta_t} L(\nabla_{\theta_t}) + \nabla_{\theta_t}\mathcal{L}_{\theta_t} \cdot \mathcal{V}_{sgd} \tag{5}$$

where $\mathcal{V}_{sgd} = \mathcal{W}_{sgd} - \frac{1}{n}1$. Wu et. al. [23] highlighted that the SGD noise can be expressed as the multiplication of the gradient matrix and a sampling noise i.e., $v_{sgd} = \nabla_{\theta_t}\mathcal{L}(\theta_t) \cdot \mathcal{V}_{sgd}$. It has been shown that the mean and covariance of gradient noise are given by:

$$\mathbb{E}[v_{sgd}] = 0 \; ; \; Var[v_{sgd}] = \frac{1}{b}[\frac{1}{n}\nabla\mathcal{L}_\theta \nabla\mathcal{L}_\theta^T - \nabla L_\theta \nabla L_\theta^T] \tag{6}$$

Based on the SGD noise formulation, a noisy gradient GD called Multiplicative SGD was proposed by Wu et al.. Given the sampling noise $\mathcal{V}$, the gradient-based parameter update becomes:

$$\theta_{t+1} = \theta_t - \eta\nabla_{\theta_t} L(\theta_t) + \eta\nabla_{\theta_t}\mathcal{L}(\theta_t) \tag{7}$$

The class of gradient noise can be controlled by deciding the class of sampling noise $\mathcal{V}$. Wu et al. studied the generalization ability of SGD for noises from different classes and found them to be on par with each other. They showed that the behaviour of Gaussian noises with Fisher covariance matrix perfectly approximates the SGD covariance noise, estimated using SVD. Here, we sample noise using Gaussian distribution with scaled Fisher covariance matrix due to its computational benefit. Inserting a structured Gaussian noise via SVD can be computationally costly due to the higher feature dimension and large number of samples, and Fisher-based covariance matrix acts as a good approximation. Wu et al. also showed that adding this compensatory gradient noise benefits the generalization of large batch training.

**Algorithm 2** Mini-Batch Multiplicative SGD

```
Result:
Initialize parameter $\theta_0$; choose learning rate $\eta > 0$, batch size $b$ and number of iterations $n$
for $i=1,2....n$ do
    Sample a mini-batch $\{x_1, x_2, ..., x_b\}$ and compute the loss $\mathcal{L}(\theta_k) = (l_{x_1}(\theta_k)....l_{x_b}(\theta_k))$
    Generate sampling noise $\mathcal{V}$ with zero mean and Fisher covariance
    Compute sampling vector $\mathcal{W}_{sgd} = 1 + \mathcal{V}$
    Compute randomized loss $L(\theta_k) = \mathcal{L}(\theta_k)\mathcal{W}_{sgd}$
    Compute gradient-update $\tilde{g}(\theta_k) = \frac{1}{B}\nabla L(\theta_k)$
    Set $\theta_{k+1} = \theta_k - \eta g(\theta_k)$
end
Output: $\theta_k$
```

#### 2.2.4 Sampling-based SGD

In mini-batch SGD, the training samples are sampled uniformly and the loss is apportioned equally among the training examples. However, all examples are not equally useful, for instance, training using redundant examples which are well represented in the dataset might not provide much benefit. Sampling-based SGD reduces the variance of stochastic gradient by biasing the selection of examples from the training set, for instance, sampling rare examples that might correspond to larger gradient updates. Traditional approaches typically leverage Importance Sampling to sample from loss distribution over each example derived froma full pass over the data. Moreover, emphasizing rare or difficult examples has been shown to reduce over-fitting [1]. Recently, Selective Backpropagation (SB) [11], a simple sampling technique which prioritizes high-loss training examples throughout training, has been shown to substantially reduce training times. By reducing computation effort on low-loss examples and the number of backpropagation passes, SB converges faster and is also able to classify difficult examples more confidently, thus improving generalization. SB leverages the idea that high-loss examples contribute more to the gradient update, thus using classification loss to derive sampling probability. Like minibatch SGD, SB also traverses the complete training set during each epoch, however, the batches are created by selecting examples with probability derived from forward loss: $P(\mathcal{L}(f_\theta(x_i), y_i))$. $P$ is computed based on the current CDF using a running set of losses of last R training samples:

$$P(\mathcal{L}(f_\theta(x_i), y_i)) = [CDF_R[\mathcal{L}(f_\theta(x_i), y_i)]^\beta \tag{8}$$

where $\beta$ determines level of selectivity.

### 2.3 Gradient descent with adaptive learning rate

One disadvantage of SGD is that it scales the gradient uniformly in all directions; this can be particularly detrimental for ill-scaled problems. Along with the problem of high gradient variance, another issue with SGD is the difficulty of tuning the learning rate. One approach to solve this problem is learning rate schedule which reduce learning rate using a pre-defined schedule based on certain thresholds. However, such pre-defined schedule is unable to adapt to dataset's charcteristics. Another issue with minimizing highly non-convex functions for neural networks is the presence of saddle points, which are surrounded by plateau of similar error values, thus making gradient close to zero in all dimensions. It is very difficult for SGD to escape from these saddle points. To correct for these shortcomings, several adaptive methods have been proposed which diagonally scale the gradient via estimates of the function's curvature. Here, we evaluate 'Adam', which is the most popular adaptive gradient optimization method in deep learning and its improved version 'ADAMw'.

**Adam** : Adaptive Momentum (Adam) computes adaptive learning rate for each parameter $\theta_t^i$ instead of using the same learning rate $\eta$ to update all parameters. This is realized by keeping a running additive sum of squared gradients per weight and dividing the learning rate by this running sum. In addition, Adam also keeps an exponentially decaying average of past gradient $m_t$, similar to momentum. The parameter update equation is given by:

$$\begin{aligned} \theta_{t+1}^i &= \theta_t^i - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} m_t \\ \text{where } \hat{m}_t &= \beta_1 \hat{m}_{t-1} + (1-\beta_1)\tilde{g}_t \\ \text{and } \hat{v}_t &= \beta_2 \hat{v}_{t-1} + (1-\beta_1)\tilde{g}_t^2 \end{aligned} \tag{9}$$

It has been shown that Adam prefers flat minima in the error curve [].

**AdamW** : AdamW modifies the implementation of weight decay in Adam, by decoupling weight decay from the gradient update. In Adam, $L_2$ regularization is implemented using weight-decay via the loss function as described in section 2.2.1. This leads to the parameter update as follows:

$$\theta^i_{t+1} = \theta^i_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}}[\beta_1 \hat{m}_{t-1} + (1-\beta_1)(\tilde{g}_t + w\theta_t)] \tag{10}$$

Since the weight decay is normalized by $\sqrt{v_t}$, weights with small and slowly changing gradients are regularized more than weights with larger gradients. Hence, $L_2$ regularization is not as effective as SGD and might lead to sub-optimal generalization. AdamW performs weight-decay only after controlling the parameter-wise step size and the regularization is thus, proportional to the weight itself.

$$\theta^i_{t+1} = \theta^i_t - \eta(\frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} + w^i_t\theta^i_t) \tag{11}$$

## 3 Experiments

Our goal is to compare the adversarial robustness of networks trained using different SGD variants and hyperparameter configurations presented in the previous section. We consider the problem of image classification and experiment with 2 network architectures: M1 [24] on MNIST dataset and VGG-11 on CIFAR-10 dataset. Model architecture for M1 is based on LeNet [15]: [Conv(5,5,20) - Conv(5,5,50) - FC(500) - SM(10)] and has been used in previous study by Yao et al. [24]. The reason behind selecting smaller architectures is due to the lower computational complexity associated with training these networks and their competitive performance observed in previous image classification benchmarking studies.

### 3.1 Datasets

In MNIST and CIFAR-10, the goal is to classify images into one of 10 classes. MNIST dataset contains 70,000 images of size 28 x 28, out of which, 60,000 are used for training and 10,000 for testing. CIFAR-10 dataset comprises of 60,000 RGB images of size 32 x 32, with 50,000 used for training and 10,000 for testing.

### 3.2 Training Configuration

Input training data was augmented with random horizontal flips for both datasets along with random cropping for CIFAR-10 dataset. The hyperparameter settings were used for training the networks using SGD and its variants are presented in Table 1. To evaluate the impact of batch-size, we increase it from 32 to 1024 in multiples of 2 for MNIST dataset. For CIFAR dataset, due to the significant training time, we perform analysis for batch size of 128. We compare minibatch SGD with other SGD-based optimization methods (highlighted previously) with a constant learning rate and zero momentum and zero weight regularization. For minibatch SGD, we vary all hyperparameters such as learning rate, weight decay, batch size and momentum and analyze their impact on classification accuracy vs adversarial robustness. For MNIST, the adversarial accuracy is evaluated using FGSM and PGD attacks on the test dataset with weaker ($\epsilon = 0.03$) and stronger attacks ($\epsilon = 0.15$). For CIFAR-10, we perform a stronger attack with higher magnitude of $\epsilon = 0.02$ for both FGSM and PGD.

### 3.3 Evaluation

We compare the various variants of SGD based on the trained-networks's adversarial accuracy for FGSM and PGD attacks on test images and the corresponding top eigenvalue of the gradient Hessian, in line with Yao et al. [24]. Since the no-attack classification accuracy can vary between different training configurations, we use attack ratio ($\frac{AttackAccuracy}{No-attackAccuracy}$) to measure the adversarial robustness. Higher eigenvalue implies sharper minimas while lower eigenvalue implies flatter minima, which in turn impacts generalization and adversarial robustness.

## 3.4 Results

### 3.4.1 Varying learning rate

**Minibatch SGD** : In this section, we firstly analyze the impact of learning rate tuning on various SGD-variants. We firstly confirm that our observations for adversarial accuracy as a function of batch size is in line with Yao et al. [24]. Figure 1 presents the result of various learning rate configurations. We observe that as batch size increases, the test classification accuracy as well as FGSM and PGD accuracy ratio decreases, indicating higher adversarial vulnerability. Eigenvalues plot correlates with the drop in adversarial robustness, with higher batch size leading to higher principal curvature. Increasing learning rate from 0.01 to 0.1 leads to higher accuracy and adversarial robustness, which is in line with the better generalization observed by [16] with higher initial learning rates. We also use an adaptive learning rate schedule proposed by Jastrzebski et al. [10], wherein the learning rate to batch size (LR/BS) ratio is kept constant. We set the ratio to 0.00015625 for SGD-based optimizers and 0.0000015625 for Adam optimizer. On training with a constant LR/BS ratio, both the classification accuracy and adversarial accuracy is maintained with an increase in batch-size. Annealing the learning rate using step schedule does not lead to a gain in adversarial robustness for MNIST, which might be due to the smaller size of the dataset with comparatively easier images to classify (Figure 5).

For CIFAR-10 dataset (Table 2), even when using a constant LR/BS ratio, the no-attack accuracies are slightly lower for networks trained using bigger batches, along with the accuracies under FGSM and PGD attacks. Using larger batch size (512) though leads to flatter minima with substantially lower eigenvalue, but the adversarial accuracy is still lower.

**SGD variants** : For MNIST, we find that minibatch SGD is a strong baseline and other SGD-variants are able to outperform SGD with a minor gain in adversarial accuracy for certain batch sizes. For smaller FGSM attacks, when maintaining a constant learning rate, Noisy SGD and SVRG slightly outperform SGD for smaller batch size ($\leqslant 64$) and very high batch size (1024) (Figure 1). We observe that Adam with constant learning rate underperforms SGD-variants(minibatch SGD, noisy SGD, sampling-based SGD) substantially in terms of adversarial robustness, in a constant learning rate regime. In constant LR/BS regime (Figure 2), Adam outperforms all approaches for smaller batch sizes and smaller $\epsilon$. However, for larger batches, Selective BackProp slightly outperforms minibatch SGD for smaller attacks. An interesting observation is with a higher PGD attack with $\epsilon = 0.15$, all variants outperform minibatch SGD. We also observe that with constant LR/BS ratio, Adam obtains flatter minimas as evident from the lower eigenvalue. Also, variance-reduction based SGD (SVRG) generally underperforms minibatch SGD in terms of adversarial robustness in constant LR/BS regime.

For CIFAR-10 (Table 2), we observe that reduced variance SGD (SVRG) leads to better no-attack and FGSM attack accuracies than minibatch SGD. It also attains a lower eigenvalue which indicates flatter minima and better generalization. Noisy SGD though attains lower eigenvalue, it underperforms minibatch SGD in terms of adversarial robustness, indicating that better generalization might not necessarily imply lesser adversarial vulnerability. Another important result to note is that although adaptive learning rate based optimization using Adam achieves the highest no-attack accuracy, it lies in a sharper minima as evident from very high eigenvalue and attains lower accuracy than minibatch SGD under both attacks.

### 3.4.2 Weight Regularization

For MNIST, weight regularization using a weight-decay of 0.001 does not lead to significant change in adversarial robustness, except for smaller batches and higher learning rates (Figure 4). Thus, regularization benefits in smaller batches when there is a higher chance of overfitting. For CIFAR-10, we observe that using weight regularization (w=0.005) while benefits the no-attack classification accuracy slightly and also reduces the eigenvalue, it does not lead to significant change in accuracy on performing FGSM and PGD attacks.

### 3.4.3 Momentum

We also observe that using momentum leads to an increase in both accuracy and adversarial robustness on MNIST as evident from Figure 3. The result for CIFAR-10 (Table 2) also highlights that

Table 1: Hyperparameter Configurations for MNIST (batch sizes = $\{2^i\}_{i=5}^{10}$) and CIFAR-10 (batch size=128)

| **Config Name** | **Details** |
|---|---|
| lr = $\eta$ [sgd-variant] | learning rate is constant ($\eta$ = 0.01, 0.1) |
| lr/bs [sgd-variant] | learning rate/batch size is constant (c = 0.005/32) [] |
| lr = $\eta$; $w$ [sgd-variant] | learning rate is constant ($\eta$ = 0.01) with regularization $w = \{1e^{-2}, 1e^{-3}\}$ |
| lr = $\eta$; $\beta$ [sgd-variant] | learning rate is constant ($\eta$ = 0.01) with momentum ($\beta$ = 0.5, 0.9) |
| lr = $\eta$ [step] [sgd-variant] | learning rate is annealed using step schedule:<br>MNIST: starting learning rate $\eta$ is halved every 10 epochs<br>CIFAR-10: starting learning rate $\eta$ is multiplied by 0.1 at epoch 60 [] |

using momentum (m=0.9) with minibatch SGD significantly improves the classification accuracy and adversarial robustness. Higher the momentum, higher is the adversarial robustness and momentum benefits lower learning rate regime (0.01) much more. However, the loss curvature does not become substantially flatter until the momentum is increased to 0.9. We also observe that best accuracy and adversarial robustness are obtained with higher learning rate (0.1) and momentum(0.9), though using momentum with higher learning rate also leads to suboptimal training for lower batch size ($\leqslant 256$).

## 4 Conclusion

By analyzing the impact of different hyperparameters such as learning rate, momentum, batch size on performance of minibatch SGD, and comparing it to other gradient-based optimization methods derived from SGD, we note the following takeways.

1. We show that large batch training leads to less adversarial robustness and suboptimal generalization, since it leads to minimas with smaller spectrum (higher eigenvalues). Maintaining a constant learning rate to batch size ratio helps solve the large batch problem and ensures consistent adversarial robustness and accuracy.

2. Using CIFAR-10 dataset, we show that there might be some scenarios in which models having higher Hessian spectrum (highest eigenvalue) can have a higher adversarial robustness (for instance, Adam (lr=0.0005) and minibatch SGD trained (lr=0.01) outperform other SGD variants inspite of having higher largest eigenvalue). However, typically in most scenarios, eigenvalue is inversely correlated with generalization quality and adversarial robustness.

3. We showcase that minibatch SGD is a strong baseline and is able to outperform other variants such as SVRG, Noisy SGD, Selective BackProp as well as adaptive learning rate based optimization methods such as Adam in terms of adversarial robustness. In constant learning rate regimes, SGD outperforms Adam significantly and in constant learning rate/batch size regime, it performs similar to Adam except for smaller batch sizes. This is inspite of Adam having a lower highest eigenvalue than SGD.

4. Increasing learning rate of minibatch SGD improves the adversarial robustness in case of MNIST, thus highlighting that it focuses on difficult and hard-to-generalize samples with higher learning rate. Compared to using SGD variants which lead to faster and better convergence such as SVRG or Sampling-based SGD, momentum is a more promising avenue and training with higher momentum of 0.9 leads to an adversarially robust model in almost all scenarios.

Our study could be useful for future research in SGD and its related variants, we believe that it brings out certain interesting insights and highlights the major role which hyperparameter tuning of SGD plays in adversarial robustness.

## References

[1] G. Alain, A. Lamb, C. Sankar, A. Courville, and Y. Bengio. Variance reduction in sgd by distributed importance sampling. *arXiv preprint arXiv:1511.06481*, 2015.

[2] R. Babanezhad Harikandeh, M. O. Ahmed, A. Virani, M. Schmidt, J. Konečnỳ, and S. Sallinen. Stopwasting my gradients: Practical svrg. *Advances in Neural Information Processing Systems*, 28:2251–2259, 2015.

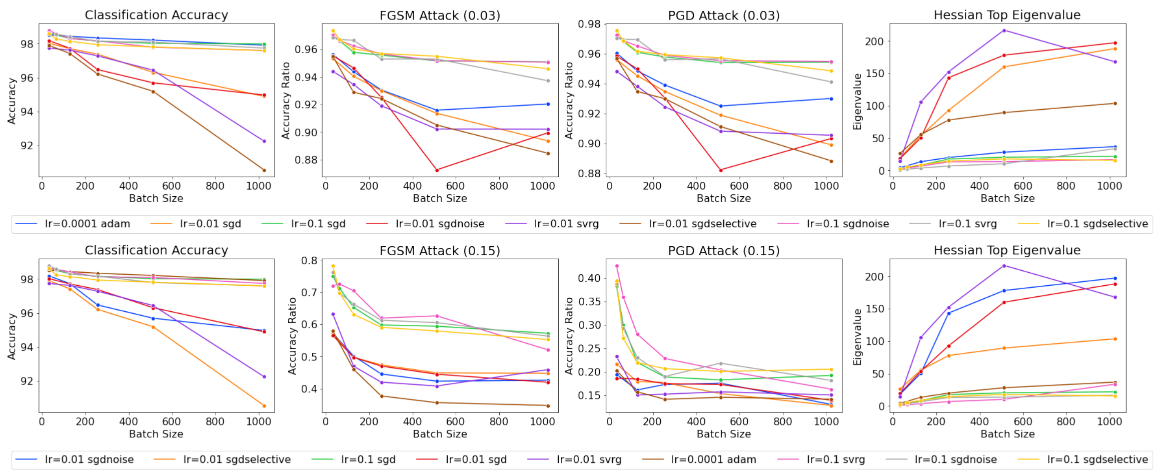

Figure 1: MNIST: SGD variants with constant learning rate (1st row: $\epsilon = 0.03$; 2nd row: $\epsilon = 0.15$)

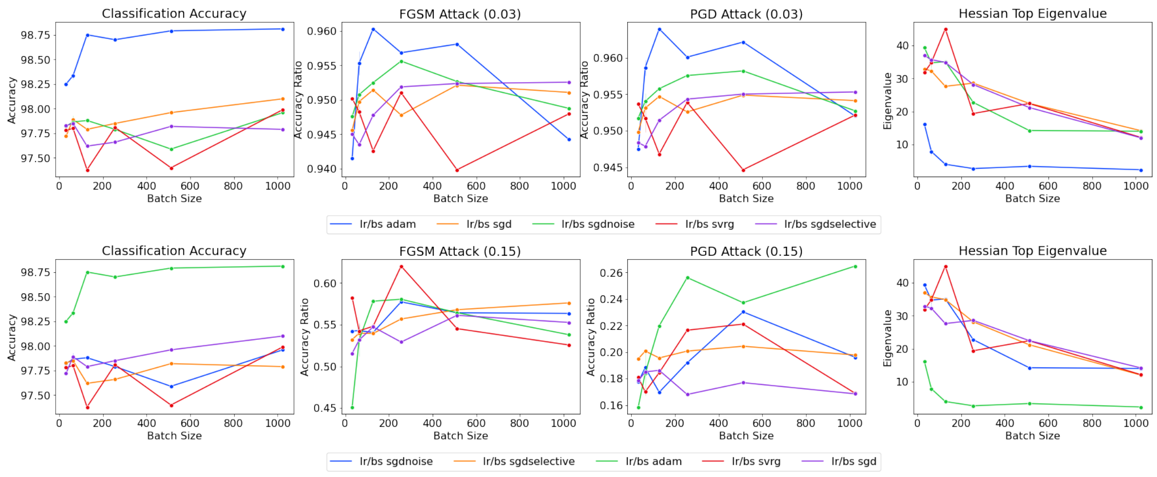

Figure 2: MNIST: SGD variants with constant LR/BS ratio (1st row: $\epsilon = 0.03$; 2nd row: $\epsilon = 0.15$)

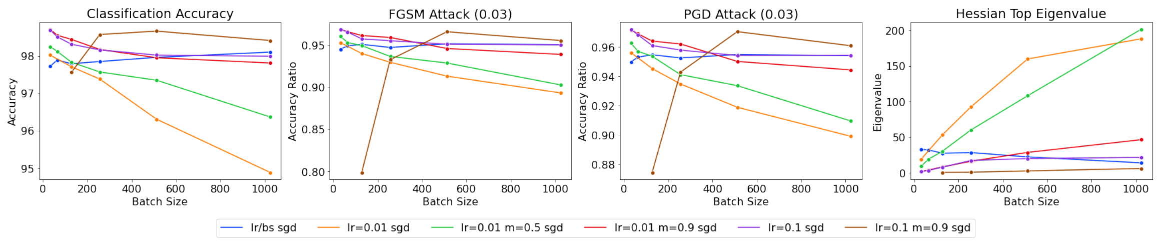

Figure 3: MNIST: Impact of momentum on minibatch SGD ($\epsilon = 0.03$)

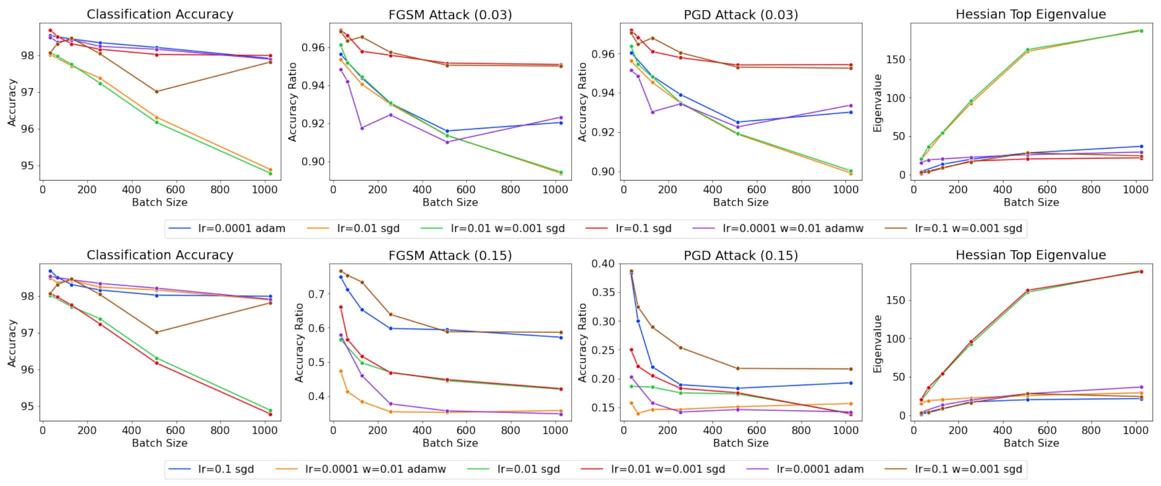

Figure 4: MNIST: Impact of weight regularization on minibatch SGD (1$^{st}$ row: $\epsilon = 0.03$; 2$^{nd}$ row: $\epsilon = 0.15$)

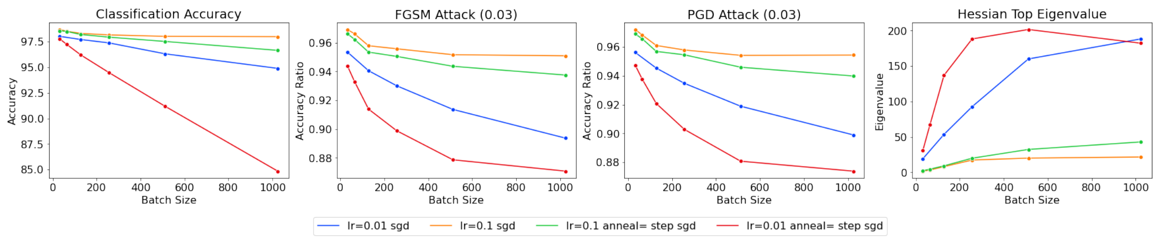

Figure 5: MNIST: Impact of learning rate annealing on minibatch SGD

Table 2: CIFAR-10: Classification accuracy under FGSM and PGD attacks ($\epsilon = 0.02$) for different variants and hyperparameters of SGD

| **Training Regime** | **Accuracy (No Attack)** | **Accuracy (FGSM Attack)** | **Accuracy (PGD Attack)** | **Hessian Largest Eigenvalue** |
|---|---|---|---|---|
| SGD Variants (constant learning rate) | | | | |
| lr = 0.1 step sgd | 86.19 | 16.82 | 38.67 | 33.39 |
| lr = 0.1 step sgdselective | 81.39 | 5.89 | 17.13 | 52.41 |
| lr = 0.1 step svrg | 86.36 | 22.61 | 34.79 | 24.88 |
| lr = 0.1 step sgdnoise | 84.48 | 13.59 | 27.27 | 16.38 |
| lr = 0.01 sgd | 77.11 | 6.28 | 15.98 | 381.3 |
| lr = 0.01 sgdselective | 63.33 | 3.77 | 10.18 | 128.41 |
| lr = 0.01 svrg | 82.14 | 2.17 | 11.41 | 198.24 |
| lr = 0.01 sgdnoise | 81.22 | 4.85 | 17.1 | 227.54 |
| lr = 0.0005 adam | 86.78 | 15.84 | 26.95 | 1863.31 |
| Minibatch SGD (constant learning rate) | | | | |
| lr = 0.01 sgd | 77.11 | 6.28 | 15.98 | 381.3 |
| lr = 0.01; m = 0.9 sgd | 86.36 | 21.72 | 38.32 | 30.21 |
| lr = 0.01; w = 0.005 sgd | 79.62 | 5.89 | 17.25 | 293.87 |
| lr = 0.01; m = 0.9; w = 0.005 sgd | 85.14 | 8.75 | 22.34 | 44.77 |
| Minibatch SGD (constant LR/BS) | | | | |
| lr/bs (bs=32) | 84.11 | 8.65 | 28.73 | 158.29 |
| lr/bs (bs=128) | 80.43 | 10.62 | 28.11 | 137.95 |
| lr/bs (bs=256) | 76.59 | 7.93 | 19.5 | 173.58 |
| lr/bs (bs=512) | 81.02 | 9.94 | 23.32 | 58.28 |

[3] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[4] D. Csiba and P. Richtárik. Importance sampling for minibatches. *The Journal of Machine Learning Research*, 19(1):962–982, 2018.

[5] Y. Deng, X. Zheng, T. Zhang, C. Chen, G. Lou, and M. Kim. An analysis of adversarial attacks and defenses on autonomous driving models. *arXiv preprint arXiv:2002.02175*, 2020.

[6] A. Fawzi, H. Fawzi, and O. Fawzi. Adversarial vulnerability for any classifier. In *Advances in neural information processing systems*, pages 1178–1187, 2018.

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

[8] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[9] E. Hoffer, I. Hubara, and D. Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in neural information processing systems*, pages 1731–1741, 2017.

[10] S. Jastrzebski, Z. Kenton, D. Arpit, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

[11] A. H. Jiang, D. L.-K. Wong, G. Zhou, D. G. Andersen, J. Dean, G. R. Ganger, G. Joshi, M. Kaminksy, M. Kozuch, Z. C. Lipton, et al. Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762*, 2019.

[12] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26:315–323, 2013.

[13] T. B. Johnson and C. Guestrin. Training deep models faster with robust, approximate importance sampling. In *Advances in Neural Information Processing Systems*, pages 7265–7275, 2018.

[14] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[15] Y. LeCun et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 20(5):14, 2015.

[16] Y. Li, C. Wei, and T. Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. In *Advances in Neural Information Processing Systems*, pages 11674–11685, 2019.

[17] S. Liu, D. Papailiopoulos, and D. Achlioptas. Bad global minima exist and sgd can reach them. *Advances in Neural Information Processing Systems*, 33, 2020.

[18] J. Ma and D. Yarats. Quasi-hyperbolic momentum and adam for deep learning. *arXiv preprint arXiv:1810.06801*, 2018.

[19] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[20] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

[21] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[22] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[23] J. Wu, W. Hu, H. Xiong, J. Huan, V. Braverman, and Z. Zhu. On the noisy gradient descent that generalizes as sgd. In *International Conference on Machine Learning*, pages 10367–10376. PMLR, 2020.

[24] Z. Yao, A. Gholami, Q. Lei, K. Keutzer, and M. W. Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. In *Advances in Neural Information Processing Systems*, pages 4949–4959, 2018.

[25] S. Ye, K. Xu, S. Liu, H. Cheng, J.-H. Lambrechts, H. Zhang, A. Zhou, K. Ma, Y. Wang, and X. Lin. Adversarial robustness vs. model compression, or both? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 111–120, 2019.