On Adversarial Robustness of Small vs Large Batch Training

Anonymous Authors[1]

Abstract

Large-batch training is known to incur poor generalization by Jastrzebski et al. (2017) as well as poor adversarial robustness by Yao et al. (2018b). Hessian-based analysis of large-batch training by Yao et al. (2018b) concludes that adversarial training as well as small-batch training leads to lower Hessian spectrum. They combine adversarial training and second order information to come up with a new large-batch training algorithm to obtain robust models with good generalization. In this paper, we empirically observe that networks trained with constant learning rate to batch size ratio as proposed by Jastrzebski et al. (2017) not only have better generalization but also have roughly constant adversarial robustness across all batch sizes.

1. Introduction

Stochastic Gradient Descent (SGD) and its variant are the current workhorse for training neural network models. Hyperparameters like learning rate, batch size and momentum play an important role in SGD for obtaining a good minimum which generalizes well. Smith et al. (2017) and Hoffer et al. (2017) have tried to study and suggest clear rules and relation between the hyperparameters. Goyal et al. (2017) show that Imagenet can be trained quickly with a nice relation only between learning rate and batch size in a distributed setting.

But there seems to be a trade-off in generalization when the network is trained with larger batches of training samples e.g. 512/1024 and above. Keskar et al. (2016) show that large batch training lead to sharp minima which is bad for generalization. They also propose a solution to handle the sharp minima issue along with better generalization. While Dinh et al. (2017) show that sharp minima on deeper networks do also generalize well. In recent work by Jastrzebski et al. (2017) have modelled SGD along with its associated hyperparameters like learning rate and batch size. They show that maintaining contant ratio between learning rate and batch size leads the gradient descent algorithm to converge to a flatter minima and this results in better generalization.

Neural networks recently have been able to give state-of-the-art results for many classification tasks but the work of Szegedy et al. (2013), Biggio et al. (2017) have exposed a serious vulnerability in neural network-based models which achieve state of the art results in various tasks like object recognition, speech synthesis, etc. These models are known to be vulnerable to small, pixel-wise changes that are almost imperceptible to the human eye, but the networks grossly misclassify the perturbed data. They obtain the small perturbation using box-constrained L-BFGS by maximizing the prediction error of the given model. Goodfellow et al. (2015) propose a quicker method based on gradients, the Fast Gradient Sign Method (FGSM) to find such an adversarial perturbation given by $x' = x + \epsilon \text{ sign}\left(\nabla_x J(\theta, x, y)\right)$, where x is the input, y represents the targets, θ represents the model parameters, and $J(\theta, x, y)$ is the cost used to train the network. Subsequent work has introduced multi-step variants of FGSM, notably, an iterative method by Kurakin et al. (2017) and Projected Gradient Descent (PGD) by Madry et al. (2018). On visual tasks, the adversarial perturbation must come from a set of images that are perceptually similar to a given image. Goodfellow et al. (2015) and Madry et al. (2018) study adversarial perturbations from the ℓ_∞-ball around the input x, namely, each pixel value is perturbed by a quantity within $[-\epsilon, +\epsilon]$.

A natural thing to do in term of obtaining an adversarial robust network would be to include the perturbed samples into the training process. This is referred to as Adversarial training. Which is an expensive step and hence, normally a mixed approach is taken where the network is trained to a good accuracy with unperturbed samples and then for a few epochs trained with perturbed samples. e.g. in Yao et al. (2018b), they first train the networks for 100 epochs with unperturbed samples and then 5-10 epochs with perturbed samples.

Obtaining a naturally robust system without adversarial training is a desirable property. This is also something which

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

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

recent work by Sabour et al. (2017) have tried to address in the architecture level without adversarial training. A recent work by Schmidt et al. (2018) try to understand the notion of adversarial generalization in terms of sample complexity and claim that a much larger sample size would be needed to achieve adversarial robustness. While Galloway et al. (2018) observe that weight decay itself can give a robust network which generalized better than robustness achieved by adversarial training.

A given network can be made robust either by explicit regularization like adversarial training, weight conditioning or by implicit regularization through hyper-parameter tuning. Our work could be viewed as understanding the influence of hyper-parameters like learning rate, batch size, momentum on the adversarial robustness of networks.

The recent work by Yao et al. (2018b) have shown that large batch training leads to networks which are less adversarially robust. They explain this by using the Hessian spectrum of the parameter space of the network and show that large batch training lead to convergence to points with high curvature. Curvature is characterized by dominant eigenvalue of the Hessian. They empirically notice a correlation between adversarial robustness of networks to curvature. This they observe with networks trained by a certain setting of hyper-parameters.

Our paper tries to understand the natural robustness of networks obtained by SGD hyper-parameter setting. We motivated by the works of Jastrzebski et al. (2017) and Yao et al. (2018b) seek to understand the relation between the network weights obtained by the setting of various hyper-parameters in SGD and its associated FGSM/PGD adversarial robustness. For this study we use MNIST, Fashion MNIST and CIFAR10 datasets. To do comparison with existing work we use M1 and C1 models from Yao et al. (2018b). We also use the network given in Table 1 which we refer to as Standard Convolutional Neural Network (StdCNN) and ResNet18 as given in He et al. (2016) as part of the study.

Our Results

We make the following important observations.

- Training the models with a constant learning rate to batch size ratio not only results in convergence to a flatter minima but also ensures that adversarial robustness does not degrade with increasing batch size.
- We show that the Hessian based analysis does not always explain the adversarial robustness in small vs large batch training.
- We show that the there are models which when trained with large batch size have higher Hessian spectrum and also better adversarial accuracy compared to small batch training.
- Adding momentum helps converge to a flatter minima by lowering the Hessian spectrum. Larger momentum values in most cases lead to a better robust model than with smaller momentum values.

2. Comparison to Hessian Based Benchmark of Yao et al. (2018b)

We first verify whether we get the same values as Yao et al. (2018b). In Figures 9, 11 for MNIST and Figures 1, 15 for CIFAR10 we plot the generalization which is the test accuracy and adversarial accuracy using FGSM attack on the test with $\epsilon = 0.3$ for MNIST and $\epsilon = 0.02$ for CIFAR10. In Figures 10, 12 for MNIST and Figures 14, 16 for CIFAR10 we plot the generalization which is the test accuracy and adversarial accuracy using PGD attack on the test with $\epsilon = 0.3$ for MNIST and $\epsilon = 0.02$ for CIFAR10. Figure 13 for MNIST and Figures 2 and 17 for CIFAR10 plot the topmost eigenvalue of the Hessian wrt to model parameters. The red lines in the figures were obtained by the exact training setting as suggested by Yao et al. (2018a) which we refer to as Benchmark in the plots. Refer to Appendix A for details of their hyper-parameter setting. Apart from this we have also trained the networks with fixed learning rate(LR) and constant learning rate to batch size ratio(LR/BS) without momentum. We plot these along with the Benchmark. The detailed analysis of these plots will be done in Section 3. For the Benchmark experiments we do observe similar values on the generalization and Hessian spectrum values as shown by Yao et al. (2018b). But for the adversarial robustness using FGSM we observe the same trend - that the accuracy drops with larger batch size. Our accuracy values are however different.

We have performed the same experiments done for MNIST on Fashion MNIST. Figures 18,3 plot the generalization, FGSM accuracy for Fashion MNIST with $\epsilon = 0.3$. Figures 19, 21 plot the generalization, PGD accuracy for Fashion MNIST with $\epsilon = 0.3$ and Figures 20, 4 for the top Hessian eigen value.

Figures 9 and 1 show that as the batch size increases the test accuracy does decrease and similarly the associated FGSM test accuracy also drops. Refer to Figures 10 and 14 for PGD results. Figures 13(top) and 2 do comfirm that with increase in batch size the curvature increases.

However, we find that their observation that increased curvature results in decrease in adversarial robustnes need not hold.

2.1. Counter Example to the Hessian based analysis of Yao et al. (2018b)

We now give examples of models which when trained with large batch have higher Hessian spectrum and also higher

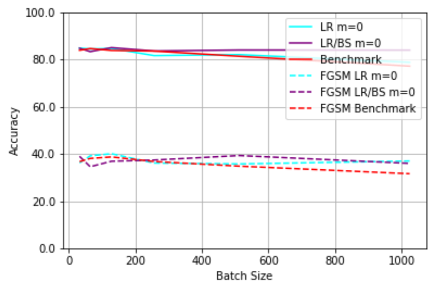


Figure 1: Test Accuracy of C1 trained with CIFAR10 and using FGSM attack with $\epsilon = 0.02$. For LR,learning rate = 0.01. For LR/BS, ratio = 0.00015625. m=momentum.

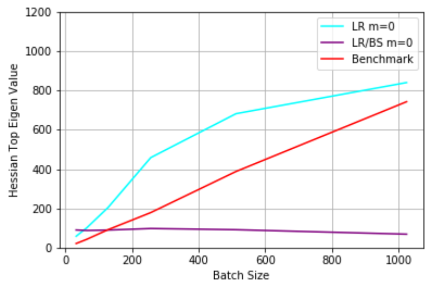


Figure 2: λ_1^{θ} : Top Eigen value of Hessian wrt to model parameters of C1 trained with CIFAR10. For LR,learning rate = 0.01. For LR/BS, ratio = 0.00015625. m=momentum.

adversarial robustness compared to small batch training. For these experiments we use the networks given in Table1 trained on MNIST and Fashion MNIST. We train the networks with a fixed learning rate. We specifically observe that in Figure 3 the FGSM accuracy increases with batch size (the light blue line in te plots), similarly in Figure 21 the PGD accuracy also increases. So this clearly indicates that increasing Hessian spectrum alone does not totally explain the change in adversarial robustness of the networks.

3. Role of Learning Rate and Batch Size ratio

Yao et al. (2018b) show that adversarial training of networks as a method to lower Hessian spectrum. Jastrzebski et al. (2017) have shown that by training a network with constant learning rate to batch size ratio the network converges to a flatter minima. We use their theory as motivation to investigate the impact of learning rate and batch size on adversarial robustness of networks.

We go about the investigation by considering various combinations of learning rate and batch size and analyse their generalization, adversarial robustness and Hessian spectrum. We use the following hyperparameter settings for

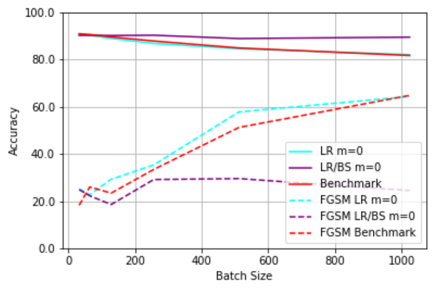


Figure 3: Test Accuracy of StdCNN trained with Fashion MNIST and using FGSM attack with $\epsilon = 0.3$. For LR, learning rate = 0.01. For LR/BS, ratio = 0.00015625. m=momentum.

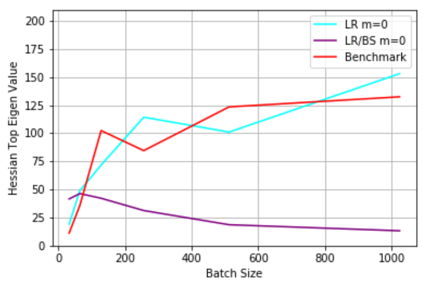


Figure 4: λ_1^{θ} : Top Eigen value of Hessian wrt to model parameters of StdCNN trained with Fashion MNIST. For LR,learning rate = 0.01. For LR/BS, ratio = 0.00015625. m=momentum.

this purpose. They are fixed learning rate(light blue), constant learning rate to batch size ratio(purple) and Benchmark (red).

We use the above hyperparameter combinations to train models M1, StdCNN on MNIST and Fashion MNIST. Models C1, ResNet18 were trained with the similar hyperparameter combinations using CIFAR10 dataset. Figures 1 to 21 contain all the plots for all the models trained with various hyperparameter setting. One major point to be noted here is that *the purple line whether its generalization, FGSM/PGD accuracy or curvature of parameter space (top Hessian eigenvalue) there is very little variation across all models and datasets.*

4. Effect of Momentum

We now analyse the role of momentum in the two setting we used to compare with Benchmark. 1) Impact of momentum with learning rate fixed and batch size changed (LR) and 2) Impact of momentum with constant learning rate and batch size ratio(LR/BS). Its clear from all the Hessian eigenvalue plots in Figures 26, 6, 30, 35, 8, 39 that momentum irrespective of whether its used with fixed learning rate or constant learning rate to batch size ratio will in most cases converge to a lower Hessian spectrum.

4.1. Effect of Momentum on LR

In the finer analysis of impact of momentum with fixed learning rate we plot training schedules with momentum values set to 0, 0.2, 0.5 and 0.9 for the models M1 and StdCNN for MNIST and models C1 and ResNet18 for CIFAR10. Figure 22 and 5 show the generalization trend and as expected with larger momentum there is better generalization. This in turn leads to better FGSM adversarial robustness as seen in 22 and 24 for MNIST and 5 and 28 for CIFAR10. For PGD robustness plots refer to 23 and 25 for MNIST and 27 and 29 for CIFAR10. Similarly in Figures 26, 6 and 30 its seen that the curvature reduces with larger momentum.

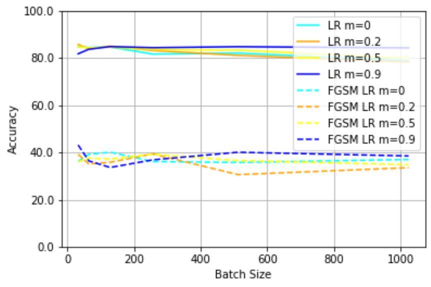


Figure 5: Test Accuracy of C1 trained with CIFAR10 and using FGSM attack with $\epsilon = 0.02$. For LR,learning rate = 0.01. m=momentum.

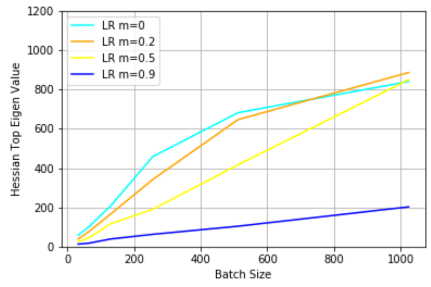


Figure 6: λ_1^θ : Top Eigen value of Hessian with varying momentum of C1 trained with CIFAR10. For LR,learning rate = 0.01. m=momentum.

4.2. Effect of Momentum on LR/BS

In the finer analysis of impact of momentum with constant learning rate and batch size ratio we plot training schedules with momentum values set to 0 and 0.9 for the models M1 and StdCNN for MNIST and models C1 and ResNet18 for CIFAR10. In Figures 35, 8 and 39 show that in most cases the curvature reduces with larger momentum, but as compared to fixed learning rate training there is no significant role of momentum with constant learning rate and batch size ratio. As the mild change in curvature or generalization does not always convert into better generalization or adversarial robustness as seen in 31, 33, 7, 37 for FGSM attack or 32, 34, 36, 38 for PGD attack.

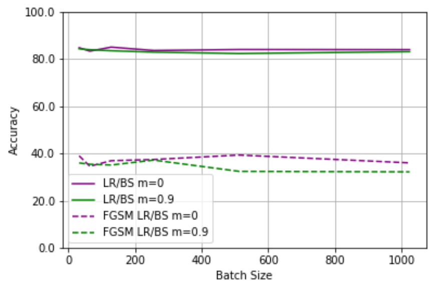


Figure 7: Test Accuracy of C1 trained with CIFAR10 and using FGSM attack with $\epsilon = 0.02$. For LR/BS, ratio = 0.00015625. m=momentum.

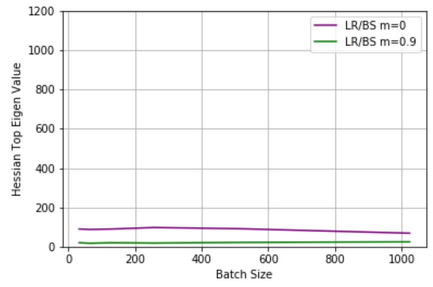


Figure 8: λ_1^θ : Top Eigen value of Hessian with varying momentum of C1 trained with CIFAR10. For LR/BS, ratio = 0.00015625. m=momentum.

5. Conclusion

We show how the modelling of SGD by Jastrzebski et al. (2017) and the Hessian spectrum can help understand the weight space and its adversarial properties. We also see how momentum plays a role in reducing the spectrum of the parameters irrespective of the ratio maintained between learning rate and batch size. We believe the paper in its current form tries to understand the role of hyper-parameters and the resultant networks robustness without any perturbed input which would be necessary to gauge the impact of adversarial training on top of it. This could aid in adapting the hyper-parameters for adversarial training.

References

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Srndic, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. *CoRR*, abs/1708.06131, 2017. URL http://arxiv.org/abs/1708.06131.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. *CoRR*, abs/1703.04933, 2017. URL http://arxiv.org/abs/1703.04933.

Galloway, A., Tanay, T., and Taylor, G. W. Adversarial training versus weight decay. *CoRR*, abs/1804.03308, 2018. URL http://arxiv.org/abs/1804.03308.

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

Goyal, P., Dollár, P., Girshick, R. B., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677, 2017. URL http://arxiv.org/abs/1706.02677.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hoffer, E., Hubara, I., and Soudry, D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *CoRR*, pp. 1731–1741, 2017.

Jastrzebski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., and Storkey, A. J. Three factors influencing minima in SGD. *CoRR*, abs/1711.04623, 2017. URL http://arxiv.org/abs/1711.04623.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836, 2016. URL http://arxiv.org/abs/1609.04836.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2017.

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

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. *CoRR*, abs/1710.09829, 2017.

Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Madry, A. Adversarially robust generalization requires more data. *CoRR*, abs/1804.11285, 2018. URL http://arxiv.org/abs/1804.11285.

Smith, S. L., Kindermans, P., and Le, Q. V. Don't decay the learning rate, increase the batch size. *CoRR*, abs/1711.00489, 2017. URL http://arxiv.org/abs/1711.00489.

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

Yao, Z., Gholami, A., Keutzer, K., and Mahoney, M. W. Large batch size training of neural networks with adversarial training and second-order information. *CoRR*, abs/1810.01021, 2018a. URL http://arxiv.org/abs/1810.01021.

Yao, Z., Gholami, A., Lei, Q., Keutzer, K., and Mahoney, M. W. Hessian-based analysis of large batch training and robustness to adversaries. *CoRR*, abs/1802.08241, 2018b. URL http://arxiv.org/abs/1802.08241.

A. Details of Datasets and Model Parameters

All experiments performed on neural network-based models were done using MNIST, Fashion MNIST and CIFAR10 datasets with appropriate augmentations applied to the train/validation/test set.

The following settings were used for training the networks. 1) LR (Smith et al. (2017)) - where learning rate is fixed to 0.01 and batch size is varied and training is done with this fixed setting for 100 epochs, 2) LR/BS(Jastrzebski et al. (2017)) - learning rate to batch size ratio is kept constant, we set the ratio to 0.00015625 and train with this fixed setting for 100 epochs, 3) setting as given by Yao et al. (2018b) where the learning rate is set to 0.01 and momentum to 0.9, and learning rate is decayed by half after every 5 epochs, for a total of 100 epochs. In 1) and 2) we do not use weight decay nor decay of learning rate. Hence, in 1) and 2) the hyperparameters - learning rate, batch size and momentum are fixed in the beginning of training with SGD and no adaptive tuning is made to the setting during the training of the networks. Any kind of weight decay or learning rate tuning was done exactly as mentioned by Yao et al. (2018b) for comparison purpose and the concerned plots are referred to as Benchmark (red line).

Data sets MNIST dataset consists of $70,000$ images of 28×28 size, divided into 10 classes. $55,000$ used for training, $5,000$ for validation and $10,000$ for testing. Fashion MNIST dataset consists of $70,000$ images of 28×28 size, divided into 10 classes. $55,000$ used for training, $5,000$ for validation and $10,000$ for testing. CIFAR10 dataset consists of $60,000$ images of 32×32 size, divided into 10 classes. $40,000$ used for training, $10,000$ for validation and $10,000$ for testing.

Model Architectures For the MNIST and Fashion MNIST based experiments we use the architectures M1 and StdCNN as given in the Table 1.

For the CIFAR10 based experiments we use the models C1 as given in Table 1 and ResNet18 architecture as mentioned in He et al. (2016). Input training data was augmented with random cropping and random horizontal flips by default.

Architectures M1 used for MNIST and Fashion MNIST experiments and C1 for CIFAR10 experiments are as given in Yao et al. (2018b) which form the benchmark for comparison.

Table 1: Architectures used for experiments

Name	Structure
StdCNN (MNIST)	Conv(3,3,10) - Conv(3,3,10) - MP(2,2) - Conv(3,3,20) - Conv(3,3,20) - MP(2,2) - FC(50) - Dropout(0.5) - FC(10) - SM(10)
M1 (MNIST)	Conv(5,5,20) - Conv(5,5,20) - FC(500) - SM(10)
C1 (CIFAR10)	Conv(5,5,64) - MP(3,3) - BN - Conv(5,5,64) - MP(3,3) - BN - FC(384) - FC(192) - SM(10)

B. Additional Plots for Section 2 and 3

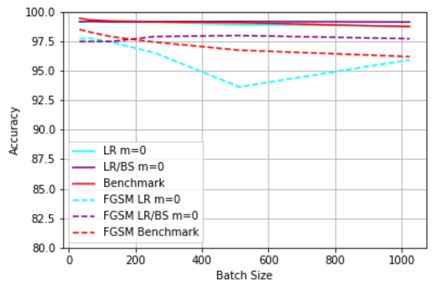


Figure 9: Test Accuracy of M1 trained with MNIST and using FGSM attack with $\epsilon = 0.3$. For LR, learning rate = 0.01. For LR/BS, ratio = 0.00015625. m=momentum.

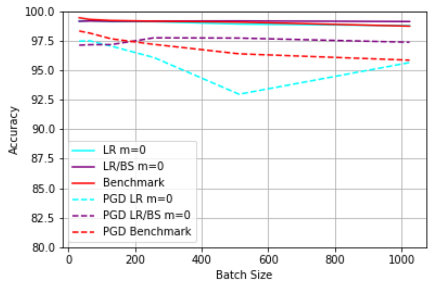


Figure 10: Test Accuracy of M1 trained with MNIST and using PGD attack with $\epsilon = 0.3$. For LR, learning rate = 0.01. For LR/BS, ratio = 0.00015625. m=momentum.

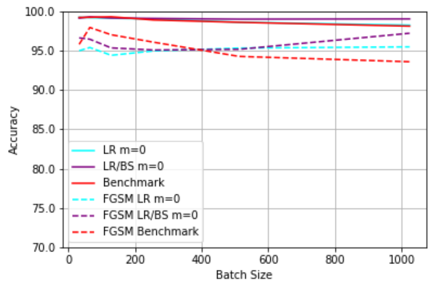


Figure 11: (left) Test Accuracy of StdCNN trained with MNIST. (right) Test Accuracy of StdCNN using FGSM attack with $\epsilon = 0.3$. For LR, learning rate = 0.01. For LR/BS, ratio = 0.00015625. m=momentum.

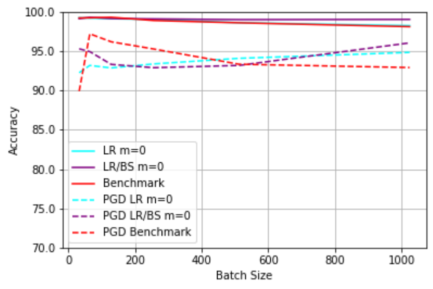
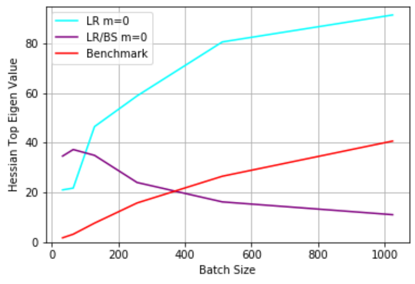


Figure 12: Test Accuracy of StdCNN trained with MNIST and using PGD attack with $\epsilon = 0.3$. For LR, learning rate = 0.01. For LR/BS, ratio = 0.00015625. m=momentum.



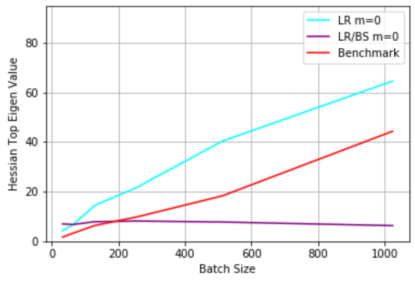


Figure 13: On MNIST, λ_1^θ : Top Eigen value of Hessian of models, (top) M1 (bottom) StdCNN. For LR, learning rate = 0.01. For LR/BS, ratio = 0.00015625. m=momentum.

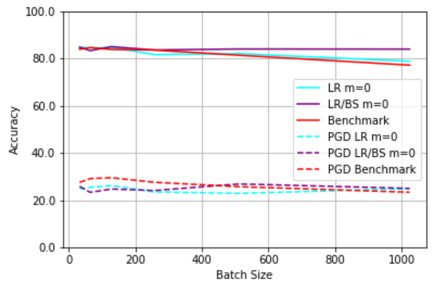


Figure 14: Test Accuracy of C1 trained with CIFAR10 and using PGD attack with $\epsilon = 0.02$. For LR, learning rate = 0.01. For LR/BS, ratio = 0.00015625. m=momentum.

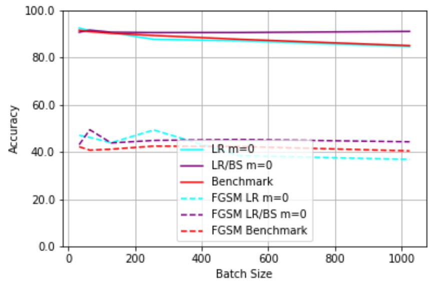


Figure 15: Test Accuracy of ResNet18 trained with CIFAR10 and using FGSM attack with $\epsilon = 0.02$. For LR, learning rate = 0.01. For LR/BS, ratio = 0.00015625. m=momentum.

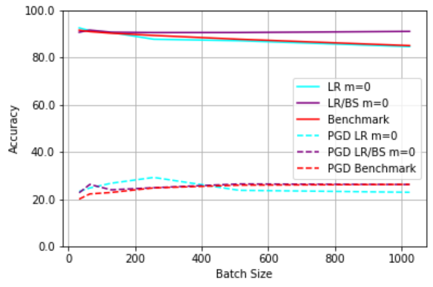


Figure 16: Test Accuracy of ResNet18 trained with CIFAR10 and using PGD attack with $\epsilon = 0.02$. For LR, learning rate = 0.01. For LR/BS, ratio = 0.00015625. m=momentum.

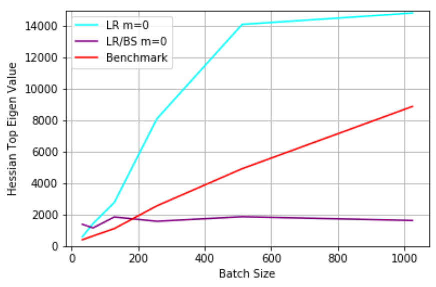


Figure 17: λ_1^θ : Top Eigen value of Hessian wrt to model parameters of ResNet18 trained with CIFAR10. For LR, learning rate = 0.01. For LR/BS, ratio = 0.00015625. m=momentum.

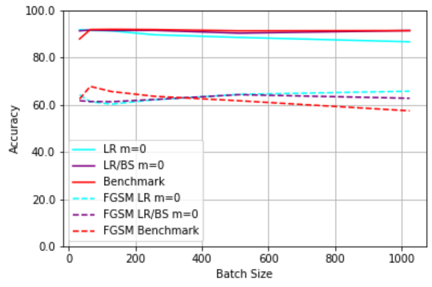


Figure 18: Test Accuracy of M1 trained with Fashion MNIST and using FGSM attack with $\epsilon = 0.3$. For LR, learning rate = 0.01. For LR/BS, ratio = 0.00015625. m=momentum.

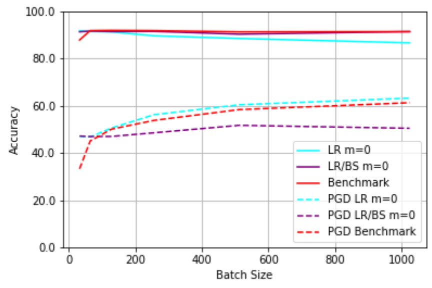


Figure 19: Test Accuracy of M1 trained with Fashion MNIST and using PGD attack with $\epsilon = 0.3$. For LR, learning rate = 0.01. For LR/BS, ratio = 0.00015625. m=momentum.

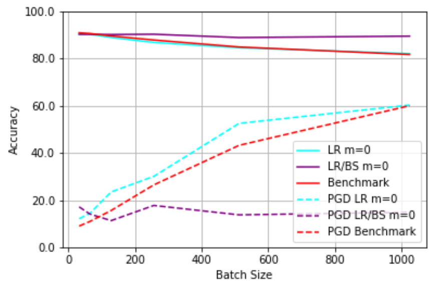


Figure 21: Test Accuracy of StdCNN trained with Fashion MNIST and using PGD attack with $\epsilon = 0.3$. For LR, learning rate = 0.01. For LR/BS, ratio = 0.00015625. m=momentum.

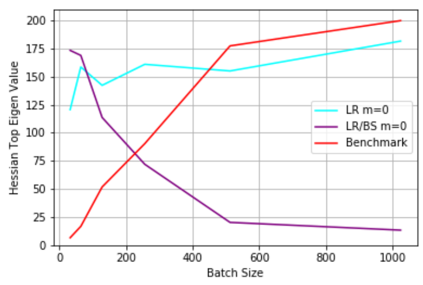


Figure 20: λ_1^θ : Top Eigen value of Hessian wrt to model parameters of M1 trained with Fashion MNIST. For LR, learning rate = 0.01. For LR/BS, ratio = 0.00015625. m=momentum.

C. Additional Plots for Section 4

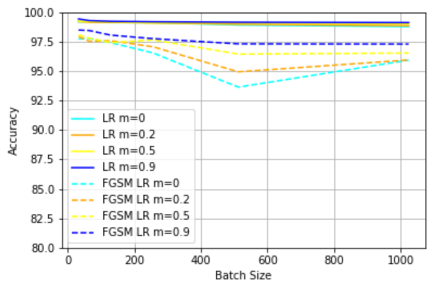


Figure 22: Test Accuracy of M1 trained with MNIST and using FGSM attack with $\epsilon = 0.3$. For LR, learning rate = 0.01. m=momentum.

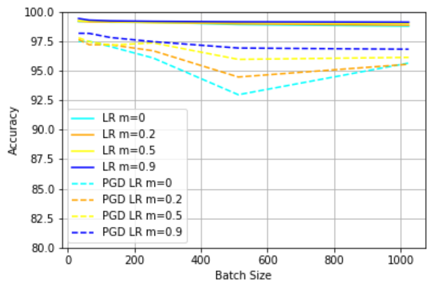


Figure 23: Test Accuracy of M1 trained with MNIST and using PGD attack with $\epsilon = 0.3$. For LR, learning rate = 0.01. m=momentum.

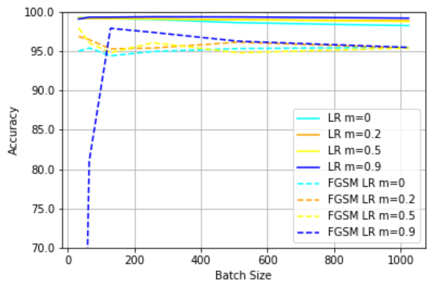


Figure 24: Test Accuracy of StdCNN trained with MNIST and using FGSM attack with $\epsilon = 0.3$. For LR, learning rate = 0.01. m=momentum.

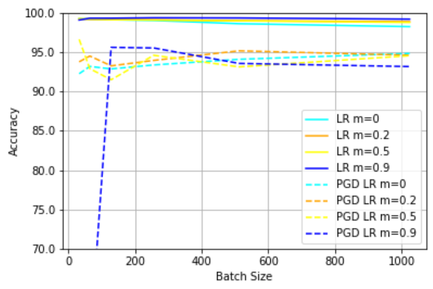


Figure 25: Test Accuracy of StdCNN trained with MNIST and using FGSM attack with $\epsilon = 0.3$. For LR, learning rate = 0.01. m=momentum.

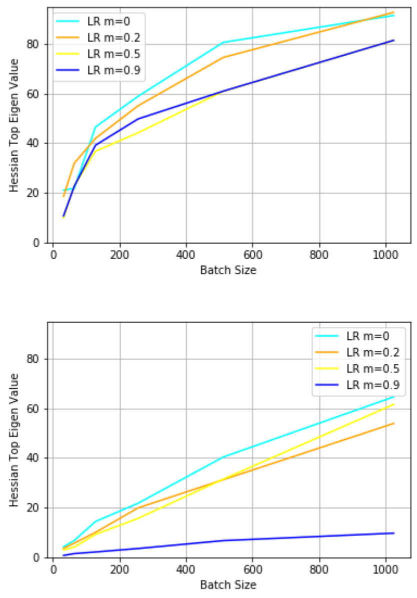


Figure 26: On MNIST, λ_1^θ : Top Eigen value of Hessian of models trained with varying momentum, (top) M1 (bottom) StdCNN. For LR, learning rate = 0.01. m=momentum.

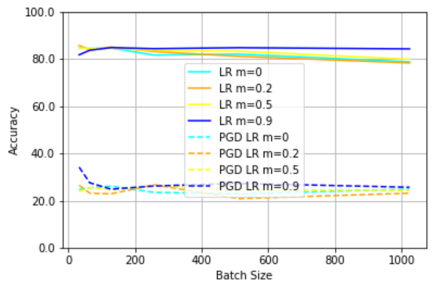


Figure 27: Test Accuracy of C1 trained with CIFAR10 and using PGD attack with $\epsilon = 0.02$. For LR, learning rate = 0.01. m=momentum.

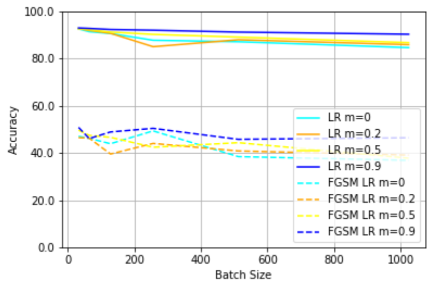


Figure 28: Test Accuracy of ResNet18 trained with CIFAR10 and using FGSM attack with $\epsilon = 0.02$. For LR, learning rate = 0.01. m=momentum.

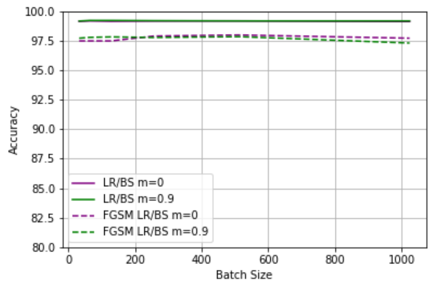


Figure 31: Test Accuracy of M1 trained with MNIST and using FGSM attack with $\epsilon = 0.3$. For LR/BS, ratio = 0.00015625. m=momentum.

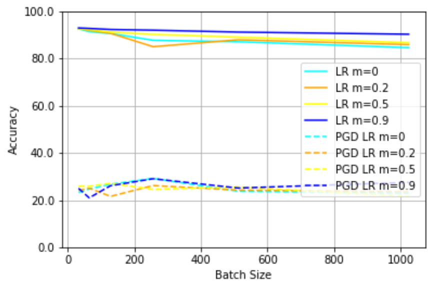


Figure 29: Test Accuracy of ResNet18 trained with CIFAR10 and using PGD attack with $\epsilon = 0.02$. For LR, learning rate = 0.01. m=momentum.

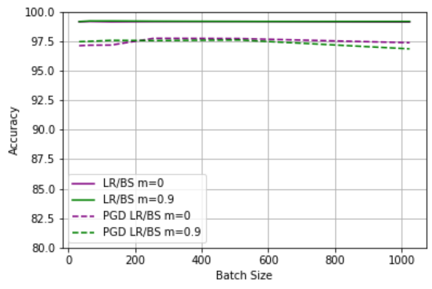


Figure 32: Test Accuracy of M1 trained with MNIST and using PGD attack with $\epsilon = 0.3$. For LR/BS, ratio = 0.00015625. m=momentum.

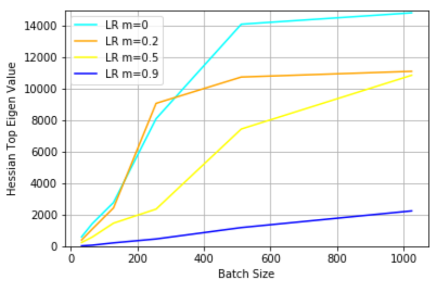


Figure 30: λ_1^θ : Top Eigen value of Hessian with varying momentum of ResNet18 trained with CIFAR10. For LR, learning rate = 0.01. m=momentum.

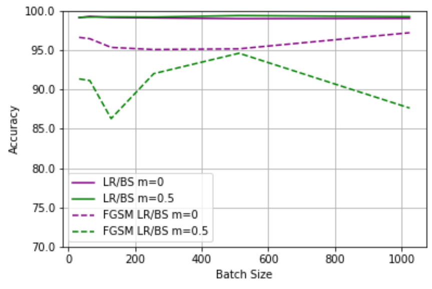


Figure 33: Test Accuracy of StdCNN trained with MNIST and using FGSM attack with $\epsilon = 0.3$. For LR/BS, ratio = 0.00015625. m=momentum.

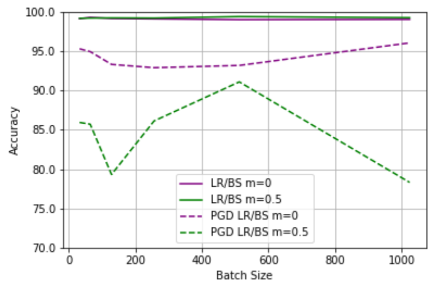


Figure 34: Test Accuracy of StdCNN trained with MNIST and using PGD attack with $\epsilon = 0.3$. For LR/BS, ratio = 0.00015625. m=momentum.

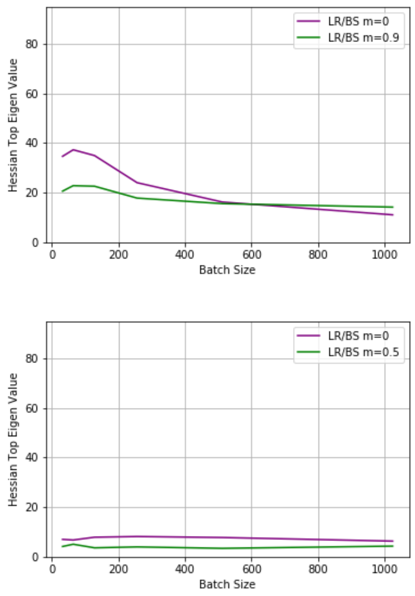


Figure 35: On MNIST, λ_1^θ : Top Eigen value of Hessian of models trained with varying momentum, (top) M1, (bottom) StdCNN. For LR/BS, learning ratio = 0.00015625. m=momentum.

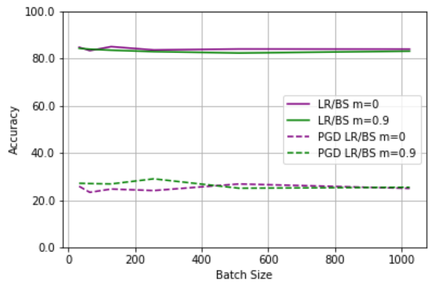


Figure 36: Test Accuracy of C1 trained with CIFAR10 and using PGD attack with $\epsilon = 0.02$. For LR/BS, ratio = 0.00015625. m=momentum.

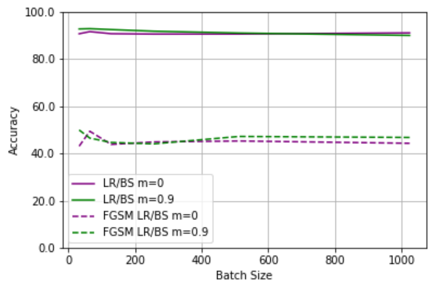


Figure 37: Test Accuracy of ResNet18 trained with CIFAR10 and using FGSM attack with $\epsilon = 0.02$. For LR/BS, learning ratio = 0.00015625. m=momentum.

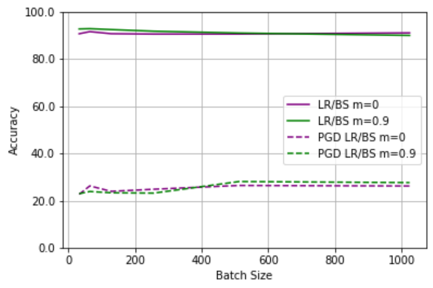


Figure 38: Test Accuracy of ResNet18 trained with CIFAR10 and using PGD attack with $\epsilon = 0.02$. For LR/BS, learning ratio = 0.00015625. m=momentum.

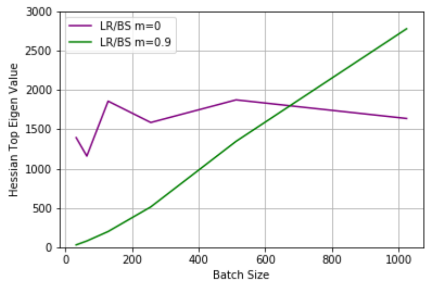


Figure 39: λ_1^{θ} : Top Eigen value of Hessian with varying momentum of ResNet18 trained with CIFAR10. For LR/BS, ratio = 0.00015625. m=momentum.