Wait, what'd ya say? - Noise Aware DNN Training

An Examination of Generalizability, Robustness, and Quantization

Authors
Ryleigh Byrne (ryleigh.byrne@duke.edu)
Wyatt Focht (wyatt.focht@duke.edu)
Code Repository

Abstract

There has been previous exploration of using SAM to improve generalizability in models. We seek to explore effective ways to improve a model's robustness against weight perturbation. This paper explores various preventative methods to attempt to mitigate the impact that weight perturbation may have on a model's performance. The preventive methods we explore include Naive noise-aware training, SAM, and SAM with multi-step weight perturbation training. Our experiments include model resistance to various strengths of weight perturbation attacks, evaluation of generalizability to new data on naive-noise, SAM, and multi-step SAM models, and evaluation of training performance of naive-noise, SAM, and multi-step SAM models.

1 Introduction

In the current literature surrounding deep neural networks (DNNs), there is a significant lack of exploration into the effect that post-training, model-weight perturbation has on the performance of a DNN. Consequently, we seek to analyze this effect by evaluating the impact that weight perturbation has on model generalizability and quantization.

In addition to this, we seek to employ various preventative methods through which we can mitigate any observed impact that weight-perturbation has on a model's generalizability & quantization and compare these results to the respective performance of a model trained with no preventative measures towards weight-perturbation.

1.1 Our Contribution

- Model-resistance to varying strengths of weight perturbation attacks
- Evaluation of generalizability to new data on naive-noise, SAM, multi-step SAM models
- Evaluation of quantization on naive-noise, SAM, multi-step SAM models
- Evaluation of training performance of naive-noise, SAM, multi-step SAM models

2 Related Works

Injecting noise or faults to DNN weights during training has been explored to promote robustness. Adding weight nose and promoting redundancy by performing aggressive weight clipping have both been shown to effectively increase DNN robustness, however, existing robustness methods can lead to a decrease in DNN performance [1]. Sharpness-aware training has gathered interest in increasing efficiency, performance, or understanding of sharpness-aware training [2]. An increase in generalization performance due to SAM inspired our experiments[3], as we sought to examine various preventative methods to mitigate the impact that weight perturbation has on a model's generalizability.

There have also been efforts to extend SAM for specific use cases. These use cases include data imbalance settings [4] and quantization-aware training [5].

Recent studies have related the sharpness of the loss landscape with robustness to adversarial noise perturbations [6]. Within this same study, robust over-fitting was examined. Robust over-fitting can be defined as having high robustness to adversarial examples seen during training, but poor generalization to new adversarial examples when testing [6]. To explore this phenomenon further, our experiments focused on robustness against noisy weights during training. In relation to this study, we attempted multiple training methods to try and achieve a model that has robustness weight-perturbation.

3 Methodology

3.1 Standard Model Training

This section serves to detail the base training parameters that are utilized while training a standard Resnet-20 model on the CIFAR-10 dataset (50,000 training images and 10,000 testing images). The following parameters were utilized:

- Epochs: 200
- Optimizer: Vanilla SGD
- Learning Rate: 0.1
- Momentum: 0.9
- L2 Regularization Strength: 1e-5

The above training specifications were determined utilizing best-practice criteria discussed both in lecture and previous laboratory experimentation.

3.2 Perturbing Weights with Gaussian Noise

The purpose of this section is to describe the mechanism through which we are able to "simulate" a weight-perturbation attack on a previously-trained model. We utilize this procedure on all models generated throughout our research. The mechanism is as follows:

1. Acquire previously-trained model
2. Access weight parameters from the model
3. Iterate over all layers' weight parameters
4. If layer is Convolutional or Linear
   (a) Generate random Gaussian noise in the same shape as the current weight's shape
   (b) Multiply noise by chosen PERTURBATION STRENGTH
   (c) Add result back to original layer weights

By following the above procedure, it is possible to generate the requisite Gaussian noise for weight perturbation. The perturbation strength was chosen as a factor of 0.1. During experimentation, perturbation strengths between 0.0 and 0.2 were tested and evaluated. When utilizing a strength of 0.2, the accuracy for the standard model was reduced to the equivalent of random guessing. Additionally, a strength of 0.0 would result in no weight perturbation. Following this, a strength of 0.1 was settled upon and utilized for all further weight perturbation results achieved in this research.

3.3 Naive Noise-Aware Training with Gaussian Noise

The method through which we are able to achieve Naive Noise-Aware Training is very similar to the mechanism described in **3.2**. It follows the standard training procedure described in **3.1** except for one difference: for every epoch during training,

1. Access weight parameters from the model
2. Iterate over all layers' weight parameters

3. If layer is Convolutional or Linear
    (a) Generate random Gaussian noise in the same shape as the current weight's shape
    (b) Multiply noise by chosen PERTURBATION_STRENGTH
    (c) Add result back to original layer weights
4. Proceed normally with standard forward and backward passes
5. Repeat for each epoch

3.4 Adversarial Noise-Aware Training with SAM

Adversarial Noise-Aware Training with SAM is implemented by following the standard model training described in **3.1** while also utilizing an additional set of steps sourced from a package found in an external, highly-cited Github repository [SOURCE HERE]. The main differences in training are as follows:

- Choice of Optimizer: This implementation creates a specialized SAM wrapper for the standard SGD-with-momentum optimizer that is used in previous model training. The neighborhood size hyperparameter, ρ, is set to 0.5 for this experiment.
- Training: The implementation of SAM utilizes two sets of forward passes, backward passes, and optimizer steps. Consequently, per epoch in training, each of these three steps are implemented twice.

3.5 SAM with Multi-Step Weight Perturbation

This method of training is similar to the methodology described in **3.4** except for the fact that ρ is gradually increased throughout training. The schema through which this is implemented entails starting with $\rho = 0.0$ and increasing ρ by a value of 0.02 every 10 epochs. By the conclusion of training, ρ will have increased to a value of 0.4. This set of parameters was largely influenced by the experimentation performed in [3] in which the research team varied ρ between 0.01 and 0.5 to find an optimal value for the smoothest convergence while training with SAM. Consequently, ρ was chosen to vary between a similar range for our experimentation.

3.6 Model Quantization

Model quantization is achieved programmatically by declaring a Resnet-20 model and specifying the Nbits hyperparameter to the desired level of quantization. After this, the previously-trained model weights can be loaded onto the initialized model. From here, the models can be tested as desired.

4 Experiments

4.1 Training

Within this section, the following set of results details further specifics of the training process for each of the respective models. Specifically, **Figure 1.0** depicts the training times for each of the created models as well as their relative training time compared to the standard model (= model training time / standard model training time).

Figure 2.0 depicts the relative training and validation loss curves for each of the models during their 200 epochs of training.

Model Type	Training Time (sec)	Ratio to Standard Time
Standard	4923	1
Naive-Noise	5374	1.09
SAM	6241	1.27
Multi-Step SAM	5862	1.19

Figure 1.0. Sample Model Training Times for 200 Epochs

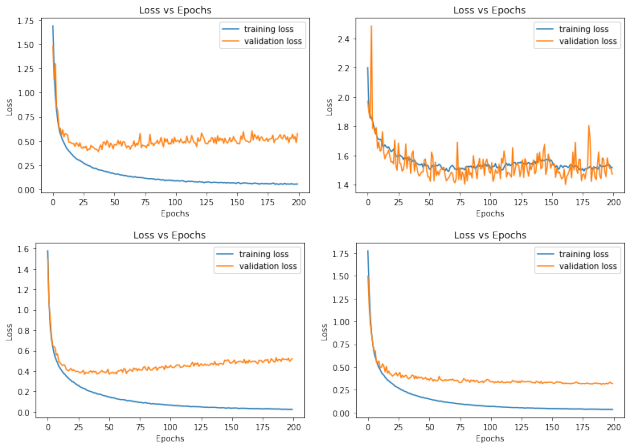

Figure 2.0. Loss Curves for Standard (top-left), Naive Noise (top-right), Standard SAM (bottom-left), and Multi-Step SAM training (bottom-right).

4.2 Generalizability

The testing accuracies displayed in **Figure 3.0** represent each fully-trained models accuracy when tested upon the 10,000 images set aside for testing.

Model Type	Testing Accuracy
Standard	0.8974
Naive-Noise	0.5031
SAM	0.9026
Multi-Step SAM	0.9109

Figure 3.0. Overall Accuracy for Each Model on Testing Data

4.3 Resistance to Gaussian Weight Perturbation

The results achieved in this section represent the relative accuracies each model achieves upon experiencing different weight perturbation attacks of varying strengths. The procedure for conducting this experiment involves loading the desired model for experimentation and following the methodology described in section **3.2** with the desired attack strength.

The x-axis of **Figure 4.0** represents the strength of the Gaussian noise that is used to perturb the model weights in this experiment. The corresponding value on the y-axis represents the accuracy of testing data that the model under examination was able to achieve given the specified level of weight perturbation.

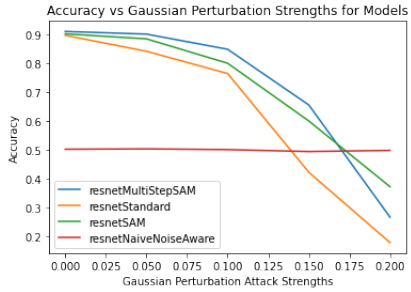

Figure 4.0. Resistance to Gaussian Weight Perturbation for Varying Attack Strengths

4.4 Robustness to Quantization

The results achieved in this section are the result of the methodology described in **3.6**. Each model is initialized with various levels of quantization and evaluated for accuracy when performing classification on the CIFAR-10 testing set.

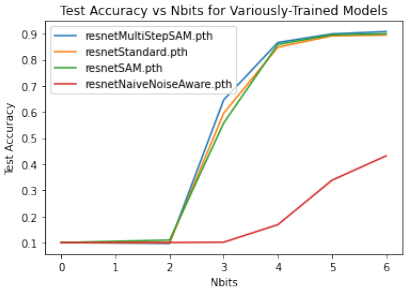

Figure 5.0. Model Accuracy under Various Levels of Quantization

5 Conclusion

5.1 Training

When analyzing training performance, the two most important features to understand are training time and training & validation loss.

5.1.1 Training Time

When analyzing each model's training time, we chose to compare each time to the base time achieved by the Standard Model, as this model was trained utilizing the typical specifications for a Resnet-20 model with no additional modifications to the training algorithm.

It can be observed that each of the other models took longer times to train than the Standard model. The Naive Noise model had the least difference in time when compared to Standard due to the fact that the training algorithm was largely the same as the Standard's with the addition of the slight computation needed to generate the random noise.

Both Standard SAM and Multi-Step SAM had significantly higher training times, as expected, due to the significant increase in computation per epoch in training (double the typical number of forward passes, backward passes, and optimizer steps).

5.1.2 Training and Validation Losses

When considering training and validation loss, the most significant difference that can be observed between the four models was that of the Naive Noise-Aware model. This difference is mostly in terms of accuracy.

For example, both the validation loss and training loss are very high for the Naive Noise-Aware model in comparison to the Standard model, Standard SAM model, and Multi-Step SAM model, as seen in **Figure 3.0**.

This effect can most likely be attributed to the extreme level of randomness that is introduced to the model weights during training. Having all model weights perturbed, even slightly, for every epoch, was bound to have a substantial impact.

5.1.3 Overfitting

Another important conclusion that can be drawn from the training experiments is that the Standard model, Standard SAM, and Multi-Step SAM models do not appear to be over-fitting, as the training loss decreases and approaches zero, but the validation loss stayed relatively consistent and did not approach zero. This could indicate that the models are not overfitting.

5.2 Generalizability

We chose to evaluate model generalizability on the ability of each model to perform accurate predictions on testing data that it had previously observed. The results of this can be found in **Figure 3.0**.

According to the results, it can be observed that the Multi-Step SAM achieves the highest level of generalizability. This is not surprising as SAM has shown previous success in both model generalization and robustness across various benchmark datasets in other works[3].

Additionally, the Naive-Noise model achieved the worst testing accuracy, which correlates with the high testing loss and validation loss observed in Section 5.1.

Acknowledgement

It should, however, be acknowledged that the levels of accuracy achieved by each of the models are slightly lower than what would be expected in a state-of-the-art implementation. This fact can be attributed to the lack of usage of a decaying training rate throughout all training algorithms implemented in this exploration. Instead, our training was intended to focus purely on decisions made during optimization in each training epoch.

In the future, we would be interested in implementing a decaying training rate throughout optimization in order to achieve accuracy closer to the state-of-the-art.

5.3 Resistance to Gaussian Weight Perturbation

5.3.1 Defining Resistance

In this context, we define "resistance" to weight perturbation as the ability of a model to maintain a relatively constant accuracy when exposed to Gaussian weight attacks of various strengths. For instance, we would say a model is resistant to Gaussian weight attacks if its base accuracy is equivalent to its accuracy under any other realistic attack strength.

5.3.2 Analysis

In terms of results, the only model that was able to achieve any significant *resistance* was the Resnet-20 model trained to be naively noise-aware of weight perturbation at each step during training (resnetNaiveNoiseAware).

To be more specific, regardless of the strength of the Gaussian-weight attack, the resnetNaiveNoiseAware's accuracy varied only slightly. However, it is important to note that although the Naive Noise model achieves the best resistance to weight perturbation, the model also achieves the worst accuracy (roughly 50% accuracy, regardless of attack strength).

In terms of accuracy, it can be observed that the other three models were able to outperform the noise-aware model for attack strengths less than 0.150. However, at perturbation strength > 0.150, the noise-aware model outperforms the standard model, and at perturbation strength > 0.175, the noise-aware model also outperforms both SAM and multi-step SAM.

5.3.3 Future Work

While the result we achieved is certainly a proof-of-concept for a possible type of training to produce a model that is entirely resistant to weight perturbation attacks, we can clearly observe a significant tradeoff when it comes to resistance and accuracy.

In the future, we would be interested in experimenting with combining the training schema of high-accuracy models (SAM and multi-step SAM) with the training schema of the naive noise-aware model. This could potentially have the effect of producing a model that is capable of not only maintaining resistance to noise attacks, but also maintaining a high accuracy simultaneously.

Following this, we would also be interested in experimenting with the level of noise that the noise-aware model is trained with as well as varying which layers of the model actually receive weight perturbation during training (beginning layers versus ending layers).

5.4 Robustness to Quantization

None of the models showed exceptional robustness to quantization, as evidenced in **Figure 5.0**. The Naive Noise performed exceptionally worse than the other three models. This could be due to this model achieving a low accuracy overall, as discussed in Section 5.2.

References

[1] David Stutz, Nandhini Chandramoorthy, Matthias Hein, and Bernt Schiele. Bit error robustness for energy-efficient DNN accelerators. Machine Learning and Systems, 2021.

[2] Jiawei Du, Hanshu Yan, Jiashi Feng,Joey Tianyihou, Liangli Zhen, Rick Siow Mong Goh, and Vincent Tan. Efficient sharpness-aware minimization for improved training of neural networks. In International Conference on Learning Representations, 2022.

[3] Foret, Kleiner, Mobahi, & Neyshabur. (n.d.). SHARPNESS-AWARE MINIMIZATION FOR EFFICIENTLY IMPROVING GENERALIZATION. T ICLR 2021. https://arxiv.org/pdf/2010.01412.pdf

[4] Hong Liu, Jeff Z. HaoChen, Adrien Gaidon, and Tengyu Ma. Self-supervised learning is more robust to dataset imbalance. In NeurIPS 2021 Workshop on Distribution Shifts: Connect- ing Methods and Applications, 2021.

[5] Jing Liu, Jianfei Cai, and Bohan Zhuang. Sharpness- aware quantization for deep neural networks. arXiv preprint arXiv:2111.12273, 2021.

[6] Xu Sun,Zhiyuan Zhang,Xuancheng Ren,Ruixuan Luo,and Liang you Li. Exploring the vulnerability of deep neural networks: A study of parameter corruption. AAAI Conference on Artificial Intelligence, 2021,

[7] https://github.com/davda54/sam.git