Deep Learning With Noise

Yixin Luo
Computer Science Department
Carnegie Mellon University
yixinluo@cs.cmu.edu

Fan Yang
Department of Mathematical Sciences
Carnegie Mellon University
fanyang1@andrew.cmu.edu

1 Introduction

Large-scale deep neutral network models have become increasingly popular to solve hard classification problems and have demonstrated significant improvements in accuracy. Due to the scale of the neutral network and the scale of the input data set, performance, in addition to accuracy, has become a significant factor in such implementations.

Recent work [1, 2] on large-scale machine learning systems propose to significantly improve the accuracy by relaxing the consistency when training neural network models (e.g. weights will not be updated every iteration). One interesting observation in these papers, alongside the above one, is that relaxation surprisingly improves the accuracy of the model.

The hypothesis of this observation (in [1]) is that relaxing consistency introduces stochastic noise into training process. This implicitly mitigates over-fitting of the model and generalizes the model better to classify test data.

Our work, taking previous works as examples and guidance, tries to study the effect of introducing different noise into different components of deep learning neural networks. We focus on introducing noise into training process of neural networks instead of into dataset. We hope that our work can provide insights into future exploration of hardware approximate computation/storage techniques to accelerate and improve the accuracy of large-scale deep learning models.

2 Problem Definition

In this project, we will investigate the hypothesis that consistency relaxing can improve model accuracy by introducing various types of noise into different neural network models.

We compare our results with recent works in [1, 2, 3, 4] and consider any improvement in accuracy or convergence rate in the final results to be a success.

3 Proposed Method

In this project, we explore three neural network models—single-layer neural network (logistic regression), multi-layer neural netwrok (mlp), convolutional neural network (LeNet).

We introduce noise into four different components in the model—weights between input layer and hidden layer, weights between hidden layer and output layer, update of gradient in logistic regression and feature mapping in convolutional neural network. The noise comes from combination of the following distributions: Gaussian, Binomial, Gamma and Rayleigh.

We experiment on three datasets—a hand-written digit dataset (MNIST), two tiny images datasets (CIFAR-10 and CIFAR-100). Specifications of datasets are summarized in Table 1.
We preprocess CIFAR-10 and CIFAR-100 by grey-scaling every image using the following formula:

$$Y = 0.2126 * R + 0.7152 * G + 0.0722 * B$$

Table 1: Datasets: MNIST, CIFAR-10, CIFAR-100

Dataset	Description	Class	Training Set Size	Testing Set Size
MNIST	hand-written digists	10	60,000	10,000
CIFAR-10	32x32 RGB images	10	50,000	10,000
CIFAR-100	32x32 RGB images	10	50,000	10,000

In other words, every pixel in the image is now a linear combination of its original RGB values. These two datasets are preprocessed due to technical implementation limitations (which will be fixed after the deadline), not machine learning theory reasons.

Our neural netwoek models are implemented using Python Theono Library. The starter code is from DeepLearning.net. Parameters of each neural network models are summarized in Table 2.

Table 2: Parameters of Neural Network Models

Model	Parameters
Logistic Regression (LR)	learning rate = 0.13
Multi-layer Logistic Regression (MLP)	LR + hidden units = 500
Convolutional Neural Network	MLP + window size = 5x5, downsample = 2x2

We use stochastic logistic regression with learning rate = 0.13. In Multi-layer Logistic Regression, there are 500 neurons in the hidden layer. During feature mapping of Convolutional Neural Network, windows are of size 5 by 5 and downsample is of size 2 by 2.

4 Experiments

4.1 Experiments on MNIST

4.2 Experiments on CIFAR-10 and CIFAR-100

5 Conclusions

References

[1] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, "Project adam: Building an efficient and scalable deep learning training system," in *OSDI*, 2014.

[2] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *ICML*, 2013.

[3] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio, "Maxout networks," in *ICML*, 2013.

[4] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *NIPS*, 2012.