AMATH 582: HOMEWORK 4

HUNTER LYBBERT

Applied Mathematics Department, University of Washington, Seattle, WA
hlybbert@uw.edu

Abstract. In this report we survey fully connected linear neural networks and the most common hyperparameters to consider for tuning. Various methods of optimization are evaluated with a variety of learning rates and momentums. Dropout and batch normalization are discussed and implemented as well. Methods of hyperparameter tuning, comparing models and results are presented. The task we are applying this to is the well known 10 class classification problem with the FashionMNIST dataset.

1. Introduction and Overview

In this report we further our understanding of machine learning concepts focusing on deep learning, the study of neural network based model architecture.

Again, our setup is a common supervised learning problem, given a collection of N data points with labels in classification or target values in a regression setting

$$\{(\boldsymbol{x_0}, y_0), (\boldsymbol{x_1}, y_1), ..., (\boldsymbol{x_{N-1}}, y_{N-1})\}.$$

The data is denoted as a matrix X and a vector of target values or class labels $\boldsymbol{y}$. We then are looking for a function f which takes in the training data and most accurately predicts the target values or class labels, written in optimization form we are looking for the following

$$f_{MLE} = \underset{f}{\operatorname{argmin}} \frac{1}{2\sigma^2} ||f(X) - \boldsymbol{y}||_2^2$$

where σ^2 is the variance of the normally distributed error terms $\epsilon \sim \mathcal{N}(0, \sigma^2)$ defined by $\epsilon_i = y_i - f(x_i)$. So said another way we are trying to minimize our errors in the classification task. The specific class of functions f to be considered to solve the problem is a Fully Connected Neural Network (FCN). We will treat the theoretical background of these methods in the next section.

Before proceeding, we would like to acknowledge the critical use of the following packages in our analysis. Namely, Matplotlib was used to create all plots and animations [1]. **TODO: Add ray tune and pytorch...!** Additionally, Scikit-learn was the primary source of using the PCA algorithm and other classification methods [2].

2. Theoretical Background

TODO: General background about the idea of neural networks

2.1. Architecture. TODO: Update

2.2. Optimizer. TODO: Update

2.3. Learning Rate. TODO: Update

2.4. Momentum. TODO: Update

Date: March 11, 2025.

2.5. Dropout. TODO: Update

2.6. Batch Normalization. TODO: Update

3. Algorithm Implementation and Development

TODO: Update

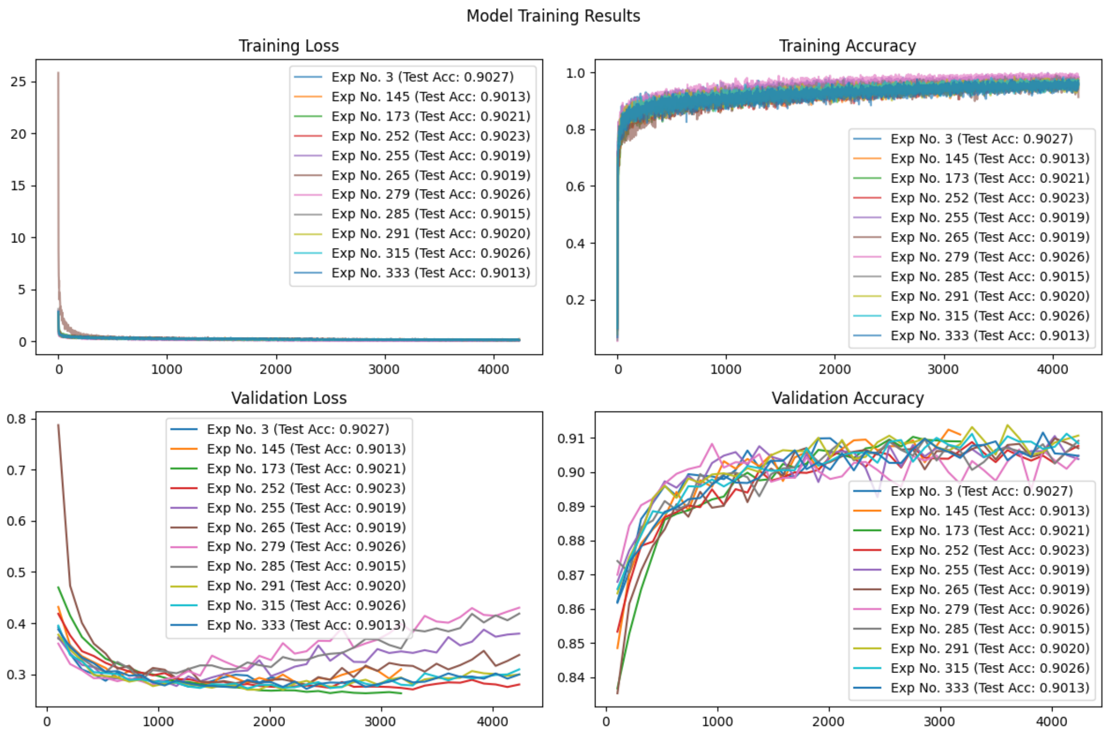

Figure 1. **TODO: Update caption**

4. Computational Results

TODO: Update

5. Summary and Conclusions

TODO: Update

Acknowledgements

TODO: Update The author is thankful to Jaxon Tuggle, Hailey Sparks, Anja Vogt, Jade Glaister, and Nate Ward for offering regular feedback and counsel when interpreting results and clarifying the implications and merits of different classifiers. We would also like to thank Professor Eli Shlizerman for carefully instructing us in class.

References

[1] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

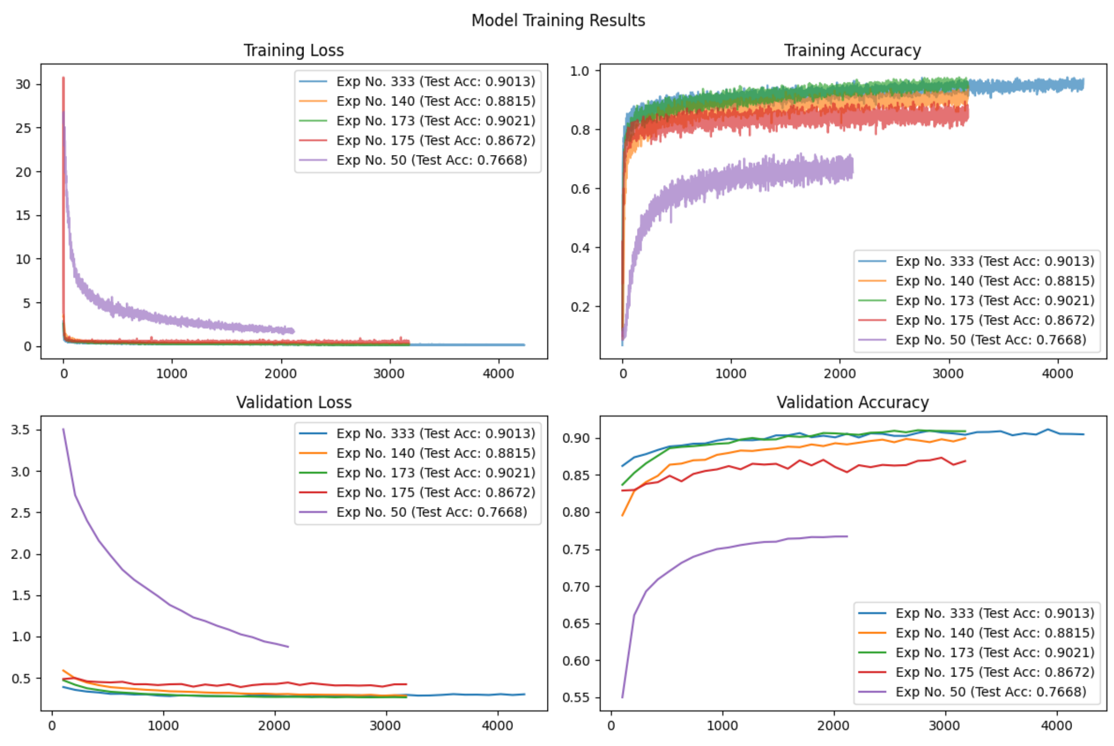

Figure 2. **TODO: Update caption**

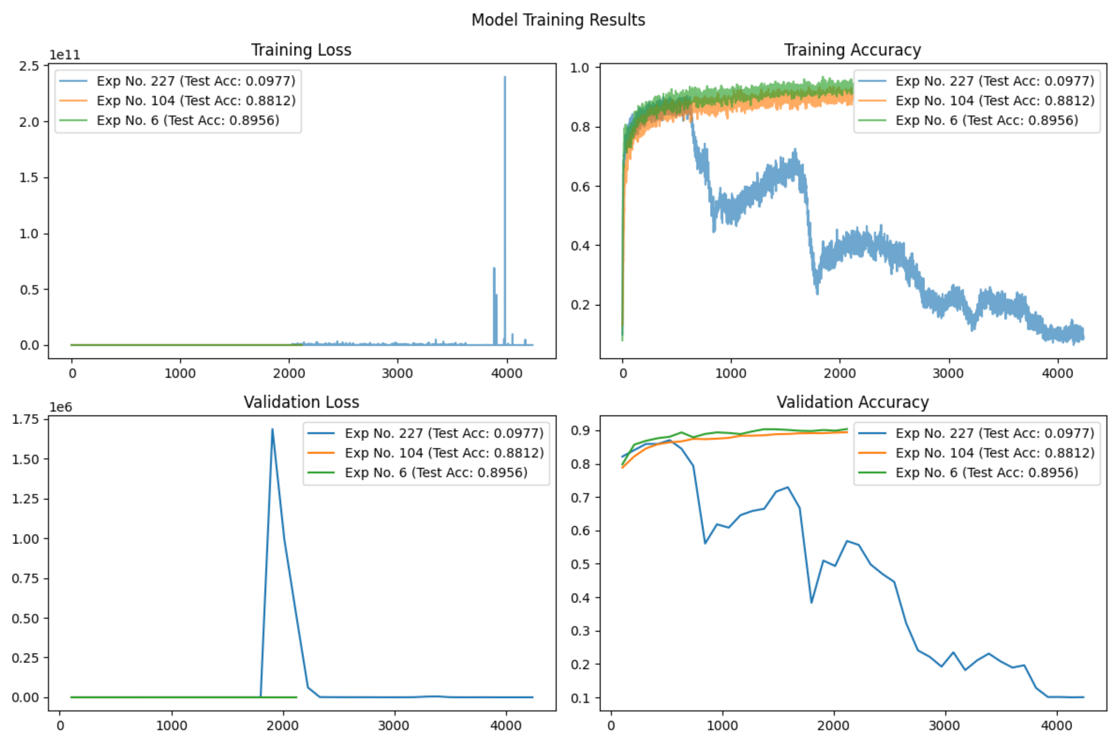

Figure 3. **TODO: Update caption**

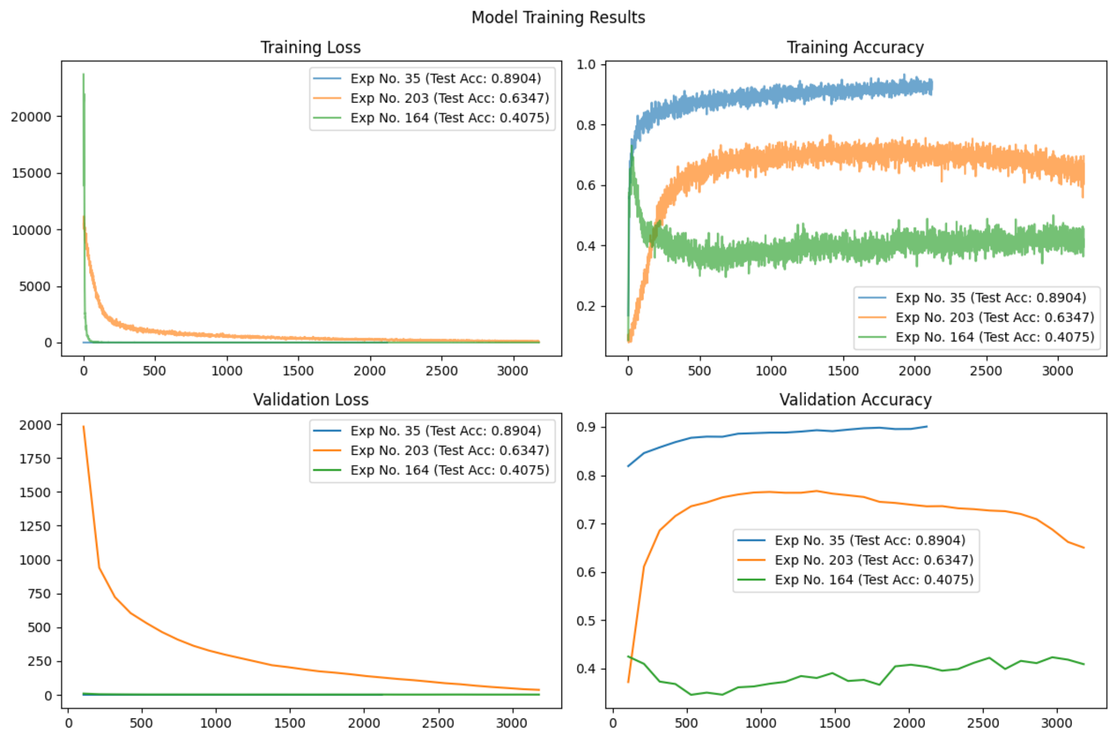

Figure 4. **TODO: Update caption**