COMP 551 Assignment 3 Write-up

Faith Ruetas, Joey Marten, Gary Zhang

December 5, 2022

Abstract

We implemented a Multi-Layer Perceptron (MLP), then ran our model on the Fashion-MNIST (Fashion) dataset. After pre-processing the data and training the model, we found that our MLP best classified the Fashion image data at rate of 88.31% with 2 hidden layers of 128 units each, a learning rate of 0.32, a batch size of 125, and 38 epochs.

1 Introduction

In machine learning projects, it is common to rely on publicly available classes such as Sklearn's MLP-Classifier. Our task was to instead implement the Multi-Layer Perceptron (MLP) model ourselves. To evaluate the performance of our MLP, we used the Fashion-MNIST (Fashion) dataset, which comprises article images of clothing items from the online retailer Zalando.

Overall, our MLP reached its maximum testing accuracy of 88.31% with 2 hidden layers of 128 units each, a learning rate of 0.32, a batch size of 125, and 38 epochs. Though we experimented to find the optimal values for each of these hyperparameters, we could not surpass Sklearn's Covolutional Neural Network (CNN) with batch normalization or Residual Neural Network (RNN) which performed at 90.21% and 90.62%, respectively.

There is ample prior work on using neural networks to classify image-based data. To spearhead innovation with tree classification, for instance, Sumsion et al found that the Multi-Layer Perceptron algorithm better classified tree genus than did vector machines and random forests [2]. Furthermore, Foody used Multi-Layer Perceptron and radial basis function models to predict land cover and crop classifications [1]. In short, our project follows pre-existing scholarship on netural networks, though with a different dataset and slightly different classification aims.

2 Dataset

2.1 Fashion

The Fashion dataset comprises images of 10 types of clothing items from the online retailer, Zalando. There are 60,000 training images and 10,000 testing images. Each instance is a 28 x 28 greyscale image with an associated integer label from 0 - 9 representing its clothing type: "T-shirt/top," "Trouser," "Pullover," "Dress," "Coat," "Sandal," "Shirt," "Sneaker," "Bag," "Ankle boot."

After loading the training and testing data into dataframes, we executed preprocessing in two ways. For one copy of the training and testing dataframes, we centered the data by performing mean subtraction before normalizing the data using the formula $(x - x_{min})/(x_{max} - x_{min})$, for each column x. For another copy of the dataframes, we normalized the data using the formula $(x - 0)/(255 - 0) = x/255$. The first method uses the column-specific minimum and maximum values whereas the second uses the data type-specific minimum and maximum values (we know the minimum and maximum greyscale value of each pixel is 0 and 255).

2.2 Analysis

To better understand the Fashion dataset, we plotted its label distributions. Both the training and testing data had an equal number of instances across each of the 10 classes (Fig 1).

We also visualized some of the images by plotting their greyscale values alongside their integer and string-value class labels (Fig 2).

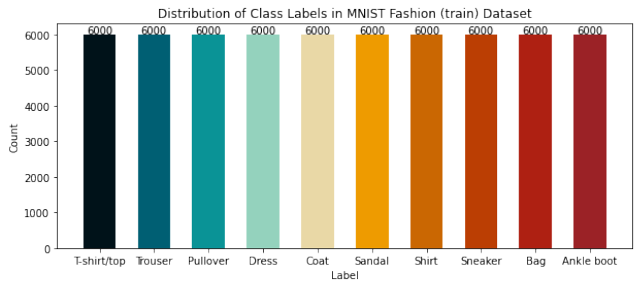


Figure 1: Both the training and testing data was split evenly among the 10 types of clothing items

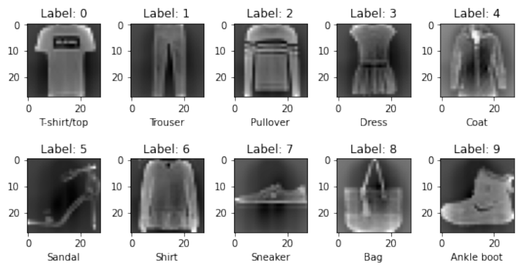


Figure 2: Visualizations of one of each clothing item based on greyscale pixel values

3 Results

3.1 MLP

Our implementation used minibatch gradient descent. It strongly reduced the maximum memory usage while fitting.

3.1.1 Baseline accuracy tests

For our first test, we experimented with a varying number of hidden layers. Layers consist of 128 units and use ReLu activation. Lastly, We tested the accuracy of model with no hidden layers, a single hidden layer, and two hidden layers. A softmax layer was added to each model to perform categorical predictions. Each model was trained for 20 epoch with a learning rate of 0.1 and a batch size of 1000. All three models performed well. The 0-layer model performed the worse with a testing accuracy

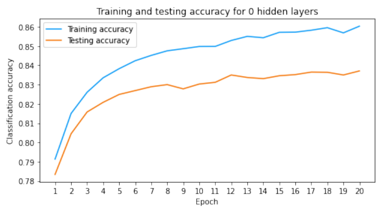


Figure 3

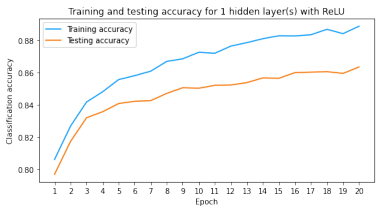


Figure 4

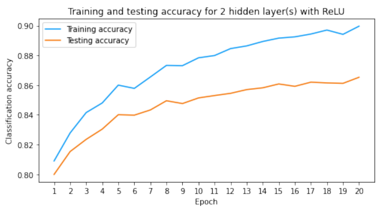


Figure 5

of **83.71%** while the 1-layer and 2-layer model performed at **86.35%** and **86.53%** respectively. We expected that the 2-layer model would perform the best due to it's higher number of parameters and non-linearity, however we did not expect the difference in accuracy to as small. The 1-layer and 2-layer perform quite similarly which could be evidence that network depth has little effect. However, non-linearity is likely the reason we saw a nearly 3% increase from the 0-layer model. The graphs show similar accuracy gains per epoch for training and testing which indicates a lack of over-fitting.

3.1.2 Activation, regularization, and unnormalization tests

	Tanh activation	Leaky-ReLU activation	With L2 regularization	Trained on unnormalized data
Training accuracy	88.76%	89.96%	89.97%	76.58%
Testing accuracy	85.48%	86.51%	86.53%	74.45%

3.1.3 Extra experiments

Here I will talk about the two extra experiments we did for MLP.

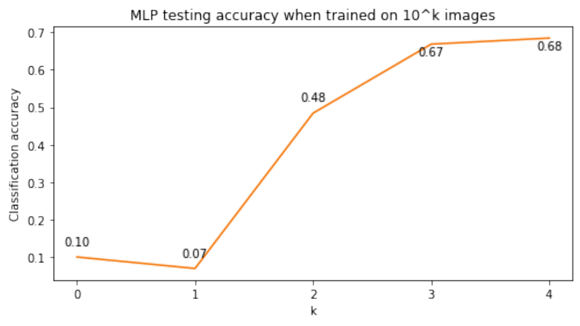


Figure 6

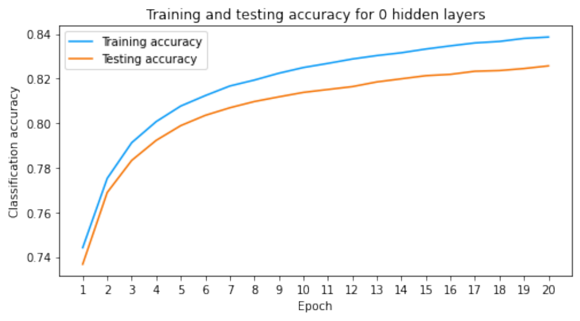


Figure 7

3.2 Finding optimal MLP hperparameters

To find the optimal hyperparameters, we tested our model on different learning rates, hidden layer/unit configurations, epochs, and batch sizes.

3.2.1 Learning rate

Hidden layers	0.001	0.01	0.1	0.2	0.3	0.31	0.32	0.33
0	24.67%	64.2%	78.34%	78.67%	76.87%	79.16%	76.96%	76.46%
1	31.45%	66.17%	79.68%	81.68%	81.97%	82.23%	82.35%	82.33%
2	36.82%	66.67%	79.99%	81.11%	82.25%	81.66%	82.74%	82.6%

3.2.2 Hidden layer/unit configurations

	[32]	[64]	[128]	[32, 32]	[32, 64]	[64, 64]	[64, 128]	[128,128]
Training accuracy	81.62%	81.72%	83.16%	78.09%	81.58%	82.38%	82.55%	83.72%
Testing accuracy	80.75%	80.12%	82.35%	77.45%	80.67%	81.26%	81.34%	82.74%

3.2.3 Epochs

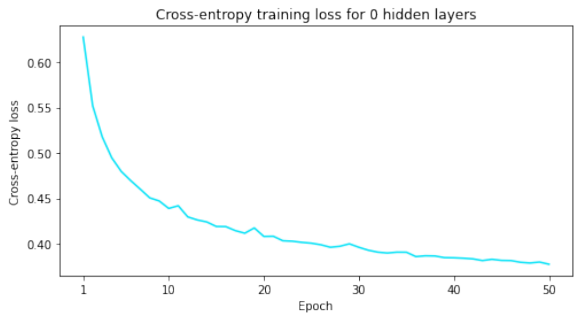


Figure 8

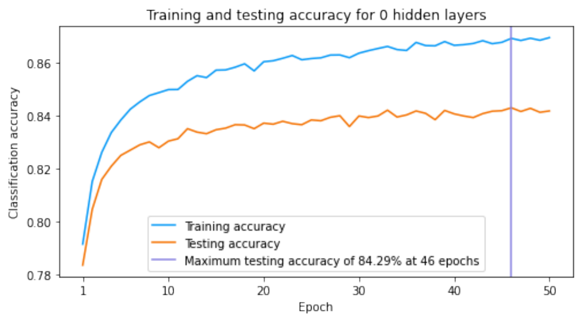


Figure 9

	100	125	150	200	500	1000
0	83.26%	83.11%	83.07%	82.1%	81.73%	80.45%
1	85.59%	85.29%	85.08%	84.81%	82.83%	81.73%
2	85.77%	85.33%	85.24%	84.89%	82.71%	81.53%

3.2.4 Batch size

3.2.5 Optimal MLP

Something something something

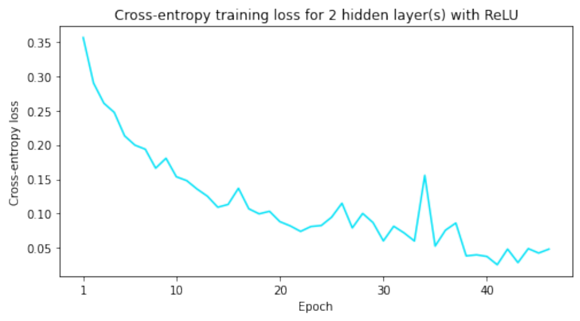


Figure 10

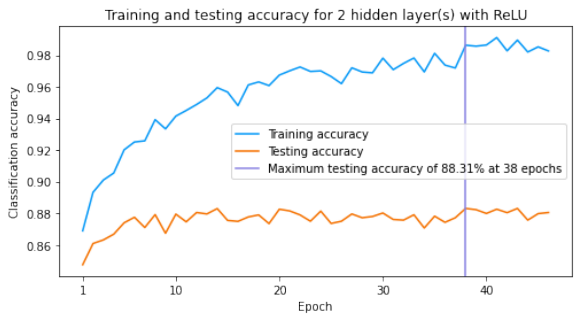


Figure 11

3.3 CNN

In our CNN model, we chose the following hyper parameters for every convolution layer: 8 filters, a filter size of 3x3, no stride, no padding, and no dilation. The convolution layers were followed by 2 dense layers each with 128 units. Lastly a softmax layer was used for categorical predictions. This model's testing accuracy out performed the 2-layer mlp by % which represents a significant amount. It's possible that the filter's captured the characteristics of the individual types better in a way that allows for a greater tolerance of different styles of the same type.

3.3.1 Baseline accuracy tests

3.3.2 Adding batch normalization

As a baseline for our next test, we chose to train a model using batch normalization. For each convolution layer, we set the activation to none and subsequently added a batch normalization layer and ReLu activation layer. The dense layer remained the same.

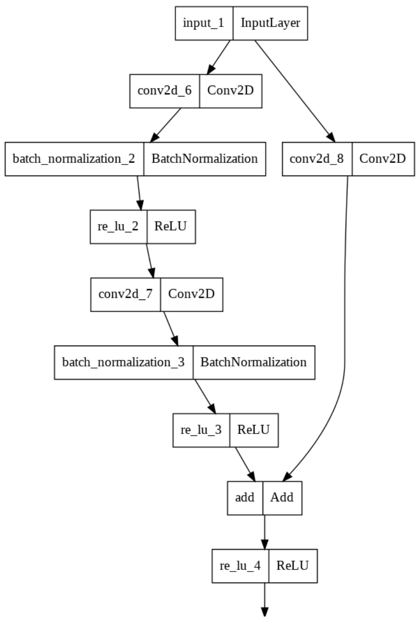
...

3.4 Residual Neural Network (ResNet)

We challenged ourselves to implement the residual block based off the structure seen in class [3]. In order to implement the ResNet, we chose to use tensor flow's functional api. A series of convolutions and batch normalizatons are applied to the input. Afterwards, the result is appended with the input that has gone through a 1x1 convolution. A 1x1 filter is applied in order to learn how much the input should be weighted. Finally it is passed to two linear layers and finally the softmax output. After training for 20 epoch with a learning rate of 0.1 and a batch size of 1000 we obtained a test accuracy of

While the improvement was marginal, the implementation challenged us and we would likely see an improvement with a deeper network and a more complex dataset.

Figure 12: Plot of our Res Block.



4 Discussion and conclusion

In summation, our 2-layer MLP model on the Fashion MNIST dataset outperformed our Multi-class Regression model on the Twenty dataset with a maximum testing accuracy of 80.62% compared to 70.22%, respectively. This difference can be explained by the fact that the latter model classified data into twice the number classes than its counterpart. Nevertheless, both algorithms outperformed Sklearn's KNeighborsClassifier on both training and testing data.

In the future, possible directions of exploration include creating deeper training models that might benefit from ResNet blocks. Additionally, it would be interesting to see the effect of max and average pooling on both the accuracy and training time due to pooling's reduction of parameters. Another direction would be an experiment in which we take a subset of the training feature data to use as unlabeled data. We could then train a semi supervised model and compare its results to our supervised model. Lastly, we think this dataset could be used in an interesting generative case. By first down-scaling the images, We could train two models to upscale them back to their original resolution. In one model we would add the label as an input. This would test if the information about the item affects quality of the up-scaling.

4.1 Statement of contributions

Faith pre-processed the data, edited code and performed experiments for MLP, and merged work on Github; Joey performed CNN and RNN experiments; Gary made functions for CNN and visualized the data as images. Everyone performed code quality reviews and edited this write-up.

References

[1] G. M. Foody. "Supervised image classification by MLP and RBF neural networks with and without an exhaustively defined set of classes". In: *International Journal of Remote Sensing* 25.15 (2004).

[2] Kimball T. Hill G. Rex Sumsion Michael S. Bradshaw. "Remote sensing tree classification with a multilayer perceptron". In: *PeerJ* 7.e6101 (2019).

[3] Yue Li. "Lecture 20 21 - Module 5.3 Convolutional neural network". In: *COMP 551 Applied machine learning* (2022), p. 36.