Neural Networks for Classifying Fashion MNIST

Jason Miller
AMATH 482

March 13, 2020

Abstract

* Given an image of a fashion item, humans are typically able to identify the object in the image easily. Neural networks can do this as well. In this assignment, we build a classifier for the Fashion-MNIST dataset, a set of images of fashion items. First we do this using fully connected neural networks, then convolutional neural networks.

1 Introduction and Overview

Our goal is to create a classifier for the fashion-MNIST dataset that performs well, as measured by the accuracy of the validation data. Practically, this means tuning the hyperparameters of our neural network until we find an optimal configuration. First, we tune hyperparameters of a fully connected neural network. Then we improve upon this arrangement by using a convolutional neural network.

2 Theoretical Background

Neural networks were inspired by how human brains supposedly work. There are layes of "neurons" with weighted edges to neurons in the layer that comes after it. There is an input layer, hidden layers, and an output layers, which correspond to the items that images are being classified as. The computations involve an activation function, such as hyperbolic tangent, bias neurons, and regularizers. Convolutional neural networks are networks that are not fully connected, i.e. each neuron does not necessarily connect to each neuron in the next layer.

3 Algorithm Implementation and Development

The process for the models for both parts that exhibited the highest accuracy on the validation data is as follows:

1. Import packages and load data.
2. Separate the dataset into 5000 images of validation data and leave the rest as training data.
3. Build the model.
4. Train the model.
5. Plot the accuracy/loss graph and look at the confusion matrix.
6. Evaluate the model.
7. Modify the hyperparameters until we optimize the accuracy.

For part 1, our final model gradually decreases in size to 10. In part 2, it starts rectangular and then gradually narrows.

4 Computational Results

There is room for improvement, but our model performs reasonably well. While experimenting with the hyperparameters, we examined the affects of adding layers to the neural network and adjusting the learning rate- both of these modifications were ultimately fruitless for us. On the fully connected network, we were able to slightly increase the accuracy by switching the activation function to hyperbolic tangent. However, for the fully connected network, the accuracy plateaued at about 89 percent. To my surprise, the convolutional network started out with worse accuracy of 76 percent, but modifying the shape to start "rectangular" with layers with uniform number of neurons, then becoming "triangular" by narrowing to 10 for the final layer increased the accuracy to about 86 percent. Notably, in part 2 the model has trouble distinguishing between T-shirts and shirts, which makes sense because I do too. As shown in figures 1) and 3), out models did not overfit the data.

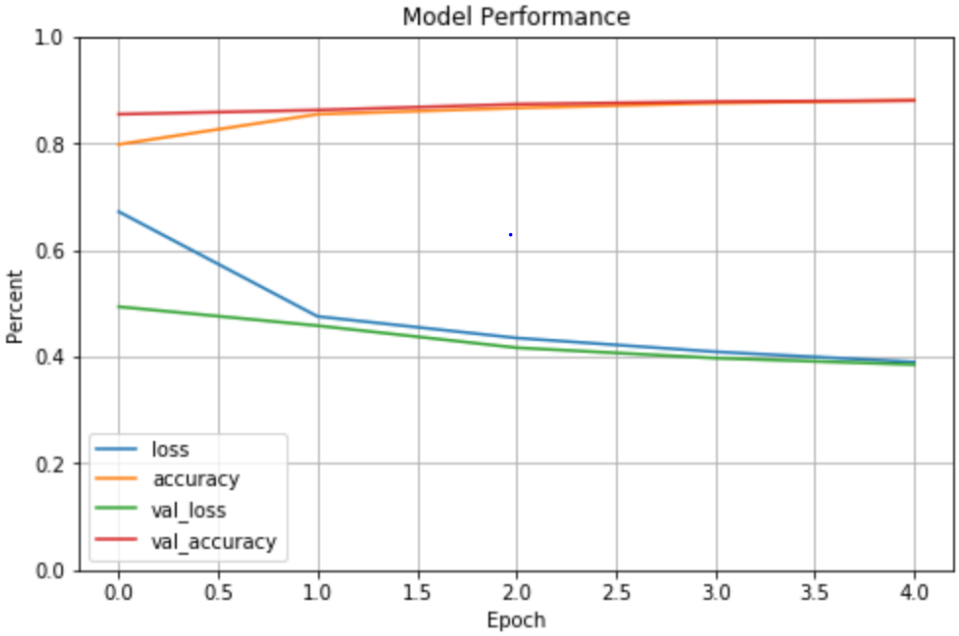

Figure 1: Model performance for part 1

```
[[4742   14  125  122   10    1  481    0   47    1]
 [  10 5320   18   72    6    1   15    0    2    0]
 [  51    4 4661   35  408    0  320    0   17    0]
 [ 175   85  101 4830  173    0  124    0   11    0]
 [   9   10  540  140 4401    0  398    0   14    0]
 [   2    0    0    1    0 5237    0  182   20   65]
 [ 669   10  555   91  297    0 3831    0   54    0]
 [   0    0    0    0    0   76    0 5242   14  156]
 [  20    3   35   15   17    5   46   14 5354    1]
 [   1    1    0    0    0   22    0  178    4 5288]]
```

Figure 2: Confusion matrix for part 1

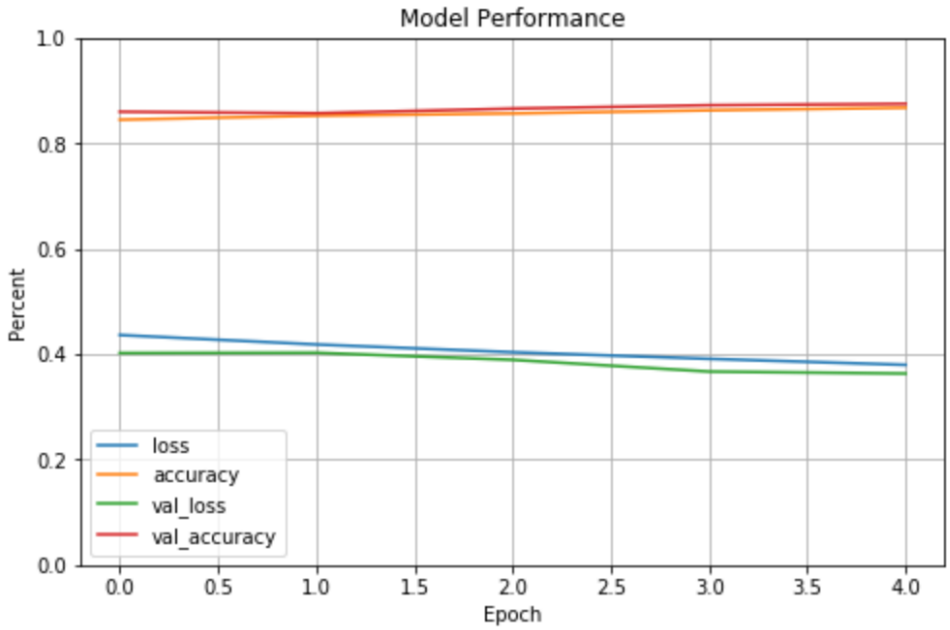

Figure 3: Model performance for part 2

```
[[4615   17  111  503   23    4  224    0   45    1]
 [  19 5185   23  193   11    1   10    0    2    0]
 [  56    7 4278   93  647    0  396    0   19    0]
 [ 141   71   41 5007  120    0  105    0   14    0]
 [  10   10  749  375 3969    2  382    0   14    1]
 [   0    1    0    4    0 5266    1  177   15   43]
 [1150   13  872  409  462    2 2553    0   46    0]
 [   0    0    0    0    0  304    0 5069    8  107]
 [  38    2   92   39   18   21  104   18 5177    1]
 [   1    0    2    1    0  103    0  275    4 5108]]
```

Figure 4: Confusion matrix for part 2

5 Summary and Conclusions

We built a classifier for the fashion-MNIST dataset that performs reasonably well in Python by tuning the hyperparameters of first a fully connected neural network, then convolutional network.

Appendix A: Python Function Descriptions

We used lots of functions and packages from Python. Namely tensorflow, keras, pandas, matplotlib, Sequential, model, partial, numpy, and sklearn.

Appendix B: Code

Part One

(Having trouble inserting this from the Jupyter notebook- will attach)