Fashionable Neural Networks

Julius C Aguma (38208345), Matt Dees (30281707), James

December 2019

1 Introduction

For our final project we decided to explore the relationship between capacity and accuracy for two types of neural networks: the feedforward neural network and the convolutional neural network. The two neural networks were trained on the Fashion-MNIST dataset [1], and thus, the input to our neural networks is one 28×28 image and the output is a classification (1-10) for that image. We measured the ratio of correct test image classifications to total number of test images as a way of measuring the strength of the neural network as capacity was varied.

2 Dataset

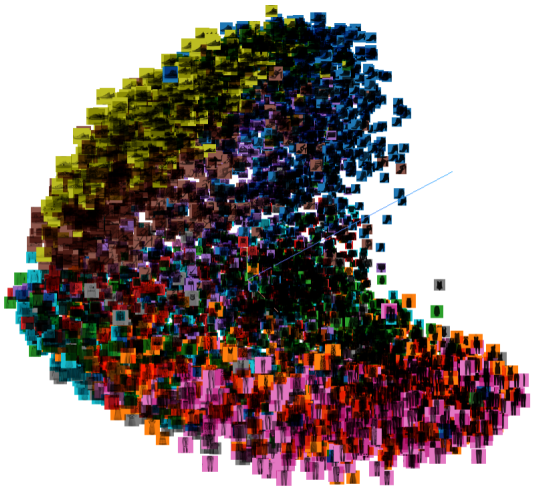


Figure 1: Tensorboard visualization of the Fashion-MNIST dataset.

We decided to use the Fashion-MNIST dataset created by Zalando Research due to the excellent documentation on Github and plethora of utilities available within their Github repository. The dataset consists of a training set and validation set. The training set contains 60,000 images while the validation set contains 10,000 images. Each image is a 28×28 grayscale image (784 total pixels). The training set is of the form $X = Y$, where X is a $60,000 \times 784$ matrix and Y is a $60,000 \times 1$ matrix. Each row in X corresponds to a different image and each column corresponds to a pixel of that image (used as features). The Y matrix is $60,000 \times 1$ matrix where each value is a classification for the corresponding row (image) in the X matrix. Each classification is an integer value which maps to one of 10 different clothing classes.

To better understand our dataset, we decided to visualize it using Tensor-

board [2] and Matplotlib [3]. Our implementation for visualizing the data is located in the **visualization.py** file in the top-level directory. We opted for a Principal Component Analysis to get a better idea of what clusters may exist amongst the data. We chose PCA because the data has a relatively high number of features (784), which is not easy to visualize. With PCA, we have access to the features the preserve the most variance in the data, and may offer some insight into how it can be organized. When loaded into the Tensorboard visualization suite we were able to observe images like in Figure 1. Once colored, it was evident that the data actually formed reasonable clusters. This led us to believe that data was reasonably separable, so we *should* see fairly high classification rates with a model using the correct features. Using Matplotlib, we were also able to visualize a singular image. Upon visualizing a single image, it was obvious that we should normalize the data. Since each pixel is a value between 0 and 255, we decided to normalize the data by dividing each pixel by 255. The proved extremely helpful in getting better results from our neural networks.

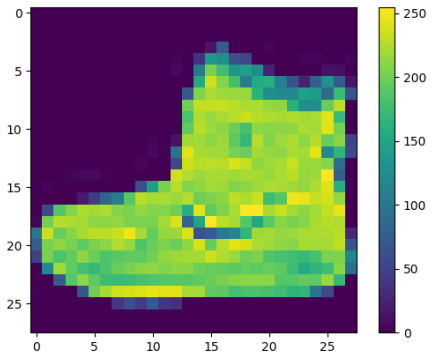


Figure 2: Image 0 from the Fashion-MNIST dataset visualized using a color map.

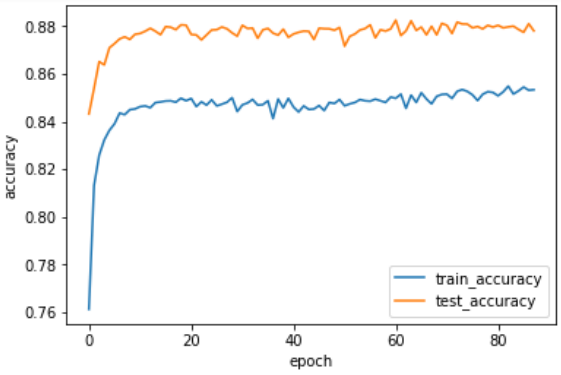
3 Neural Network Models

To analyse the capacity of neural network models on the fashion-mnist data, we chose to use a feedforward and convolutional neural network. While holding all other parameters constant, the number of hidden layers, k was varied for values $1, 2, 4, 8$. Results of this analysis are presented below with a summary of the architectures of both models.

3.1 FeedForward

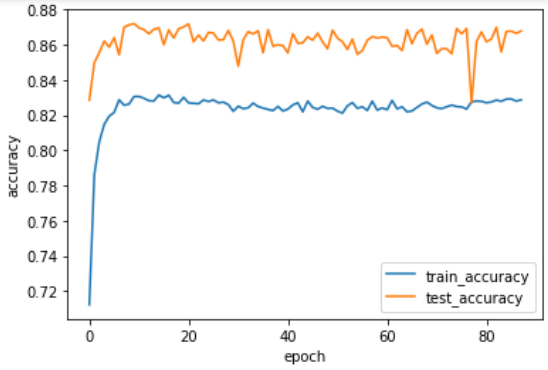
3.1.1 Architecture

Using tensorflow with keras, we built a feedforward network that takes fashion mnist data as input read through the mnist-reader.py script. The network has k hidden layers where k is varied for an analysis of capacity, that is a comparison of deep vs shallow networks on the fashion-mnist dataset. The data is split into training and validation data as shown in the model parameters below and the model is trained for 50 epochs where each epoch has 40000 iterations. In addition we use dropout factors less than 1 for regularization and cross entropy as loss function with the adadelta optimizer as a learning rate method for our gradient descent. The hidden layers use relu for an activation function while the output layer uses softmax. All inputs are normalized. Below are parameters and results for each k in the set $1, 2, 4, 8$



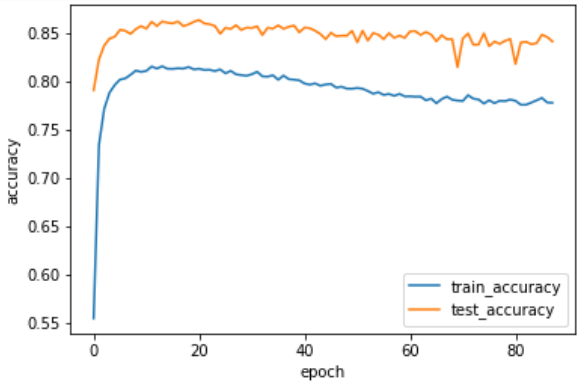

```
10000/10000 [==============================] - 0s 37us/step
Out[2]: [0.40412045266628266, 0.8688]
```

Figure 3: Feedforward neural network with $K = 1$.



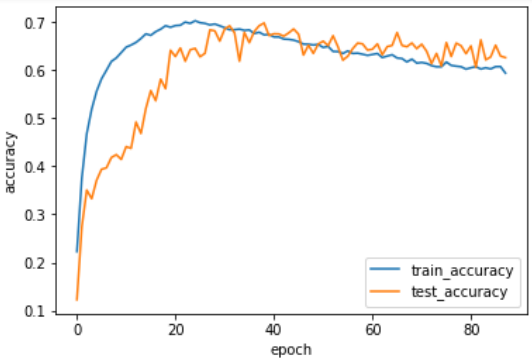

```
10000/10000 [==============================] - 0s 35us/step
Out[4]: [0.46054650063514707, 0.8587]
```

Figure 4: Feedforward neural network with $K = 2$.




```
10000/10000 [==============================] - 1s 54us/step
Out[5]: [0.549263471364975, 0.8291]
```

Figure 5: Feedforward neural network with $K = 4$.



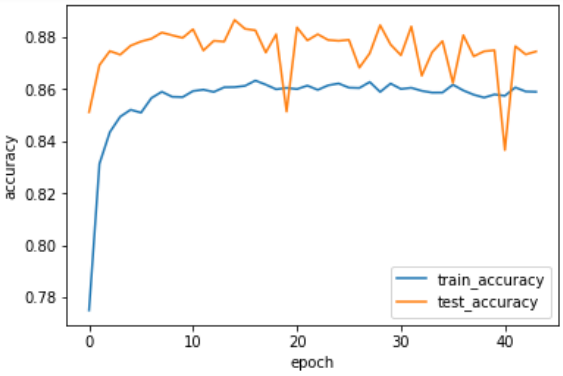

```
10000/10000 [==============================] - 1s 67us/step
Out[6]: [1.0562932844161987, 0.6197]
```

Figure 6: Feedforward neural network with $K = 8$.

3.2 Convolutional model

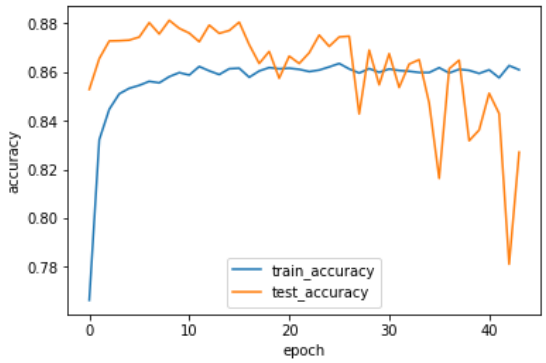
3.2.1 Architecture

The convolutional model(cnn) is similar to the feedforward model in number of hidden layers, optimizer, regularization, activation functions, and validation. However, the cnn also has 2 3x3 convolutional filters and a 2x2 max pooling layer for downsampling the convolution layers. Results follow below.



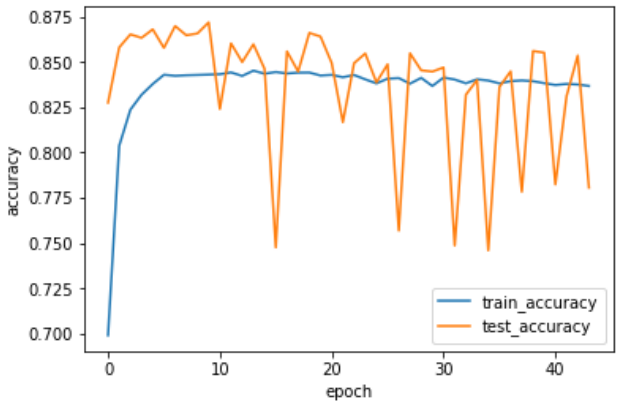

```
10000/10000 [==============================] - 2s 218us/step
Out[5]: [0.40157343983650207, 0.8689]
```

Figure 7: Convolutional neural network with $K = 1$.




```
10000/10000 [==============================] - 4s 440us/step
Out[3]: [0.5941237535476684, 0.8212]
```

Figure 8: Convolutional neural network with $K = 2$.




```
10000/10000 [==============================] - 5s 460us/step
4]: [0.7686937316894531, 0.7717]
```

Figure 9: Convolutional neural network with $K = 4$.

3.3 Discussion of results

The feedforward model gave more consistent predictions with a smoother curve while the cnn gave us higher accuracy with a peak of about 90%. For both architectures, the rate of overfitting increased with increase in the number of hidden layers. More so for the cnn where by k = 4 we see an overwhelming amount of overfitting. We also see that the peak accuracy never increased

with increase in number of hidden layers. Further proving the known fact that capacity of a neural network doesnot increase with deeper networks but only the type of functions that can be estimated improves. A good next step in the capacity study would be to look at how the network performance changes with wider hidden layer, that is hidden layers with more neurons.

References

[1] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[2] Tensorboard. `https://github.com/tensorflow/tensorboard`.

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