Network Architecture

- 2-dimensional convolutional layer with 64 filters and a 4×4 kernel
- 2×2 max pooling layer
- fully connected layer with 256 neurons and ReLU activation
- fully connected layer with 128 neurons and ReLU activation
- output layer with 10 neurons and softmax activation

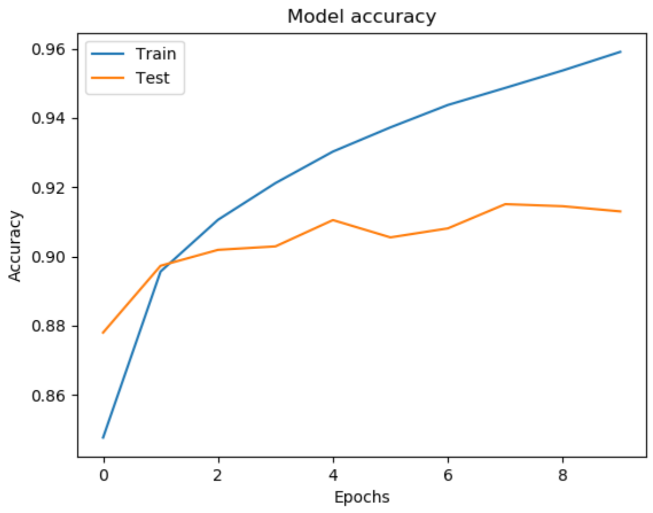

Figure 1: Train and test accuracy over number of epochs.

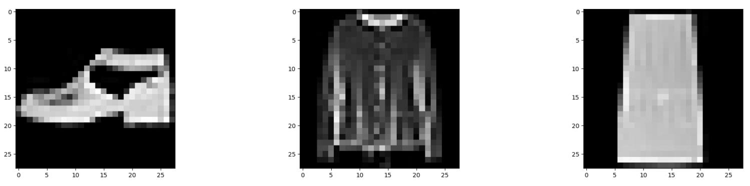

Figure 2: Misclassified examples.

Performance The model is trained for 10 epochs. The train accuracy can reach 96% while the test accuracy can reach 91.2% (Figure 1). This dataset is harder to classify than MNIST. Bias values tend to be around 0, but the weight values are not all 0 (Figure 3).

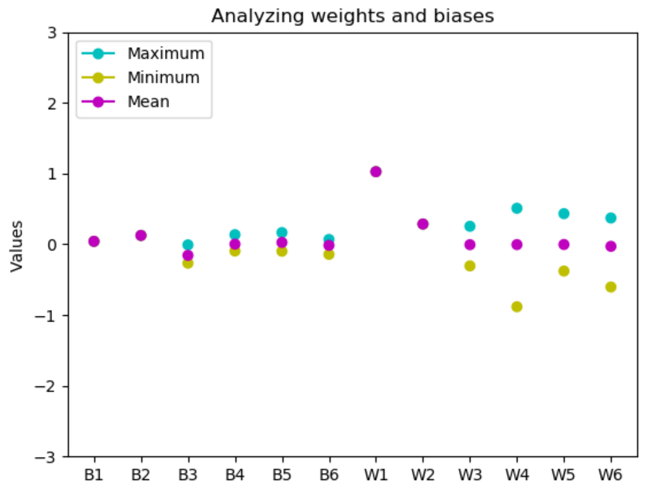

Figure 3: Final weights and biases of trained model. Each weight and bias has 3 points associated with it: maximum, minimum, and mean. These values indicate the largest, smallest, and average of values in the matrix/vector.

Figure 2 shows misclassified examples in the test set. The left image is supposed to be labeled "ankle boot", but our model predicts "sandal" instead. The center image is labeled "coat", but our model predicts a "pullover" instead. The right image is labeled "dress", but our model predicts a "shirt" instead. For these examples, I myself could not classify them correctly. So, it is not surprising that the network have misclassified these test images.

Explanation The motivation for using a CNN is that we are classifying an image dataset. Similar to the MNIST network, this network can detect features using the convolutional layer and introduce nonlinearity with the ReLU activation function. I have increased the number of filters and kernel size to extract more features from each pixel. Dropout is used again to prevent overfitting. I also have tried batch normalization which seemed to increase accuracy.

I have added an extra fully-connected layer with 256 neurons to capture more nonlinearity. It seems like the Fashion MNIST images are harder to classify, so more nonlinearity may help determine the classification boundary between the classes. The last layer also uses softmax function to output probability distribution over the classes. The loss function is again categorical cross-entropy. This network only took about 15 minutes to train on a CPU.