CS4341 - Introduction to Artificial Intelligence

Project 3: Fashion MNIST Dataset Report

Group 7: Ardavasd Ardhaldjian, Eri Kim, Hoang Phan Pham

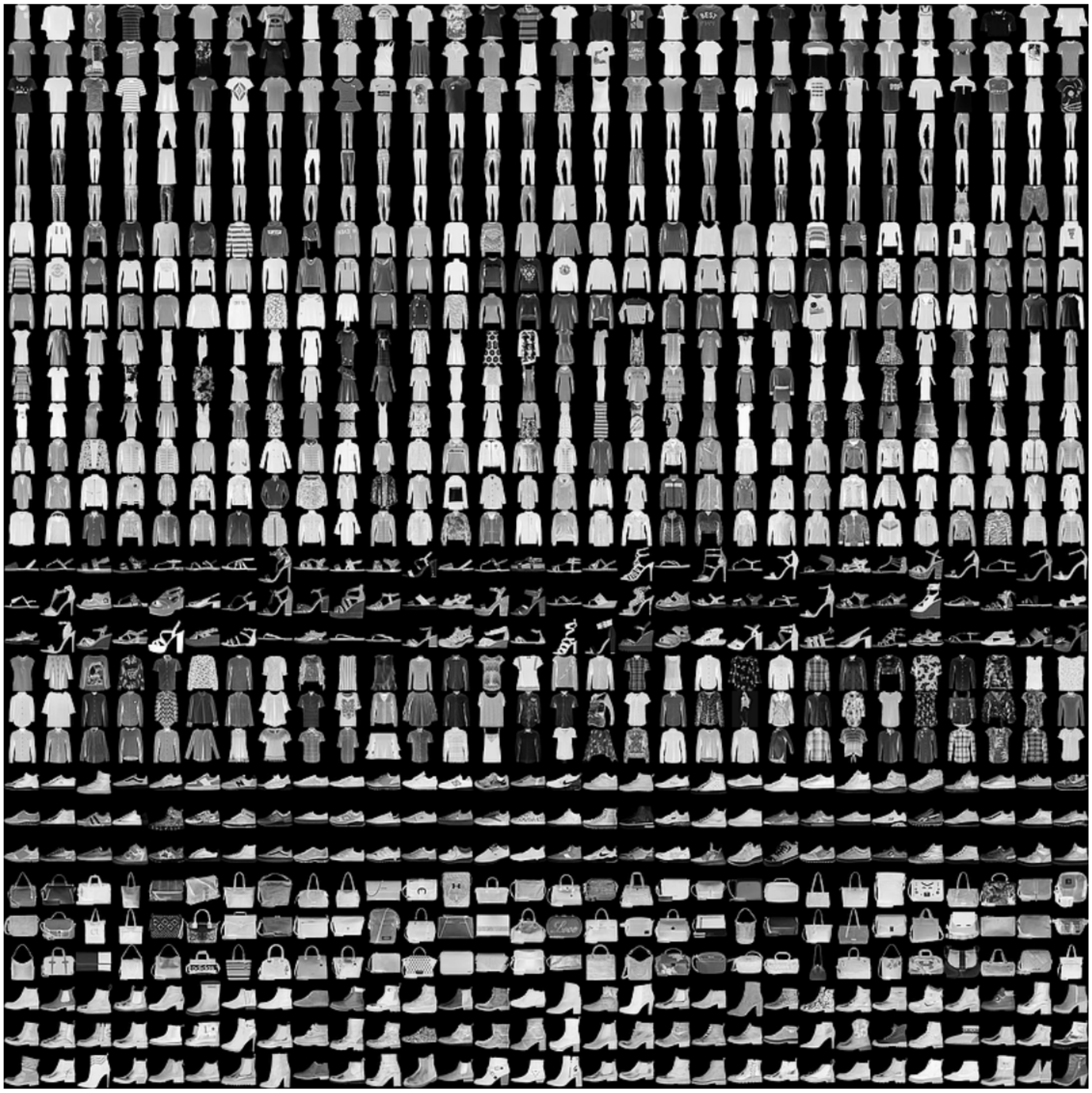


Table of Contents

0. Introduction (BAE)

The main focus of this project was to use Artificial Neural Networks (ANNs) to categorize images. Our program takes an image of a fashion item and outputs the label (0-9) that was written with a given template utilizing the Sequential model within the Keras package.

This report consists of four major parts as follows:

1. Model & Training Procedure Description
2. Model Performance & Confusion Matrix
3. Training Performance Plot
4. Visualization

1. Model & Training Procedure Description(B)

A well-built classification model requires experimentation on various hyperparameters. These parameters include activation functions, number of neurons, batch size, and more. Our group graphed the model accuracy and loss functions while testing different epoch counts and gathered that a well-performing number was 100. Similar to the epoch count, we determined the mini-batch size by running multiple models with various numbers; and the results showed that everything below and above 128 performs worse than the number itself.

Hidden layers were more tricky to experiment with, as many different combinations were performing similarly. We tried to include all of the activation functions mentioned in the project description (*ReLu, SeLu, Tanh*). However, the model performed worse with the *Selu* activation unit than without it; therefore, we have decided to remove *Selu*. Then we tried many different combinations of the *ReLu* function and various kernel initializers and found out that *Random Uniform* worked the best. In addition, *Glorot* kernel initializer performed the best with the *Tanh* activation function.

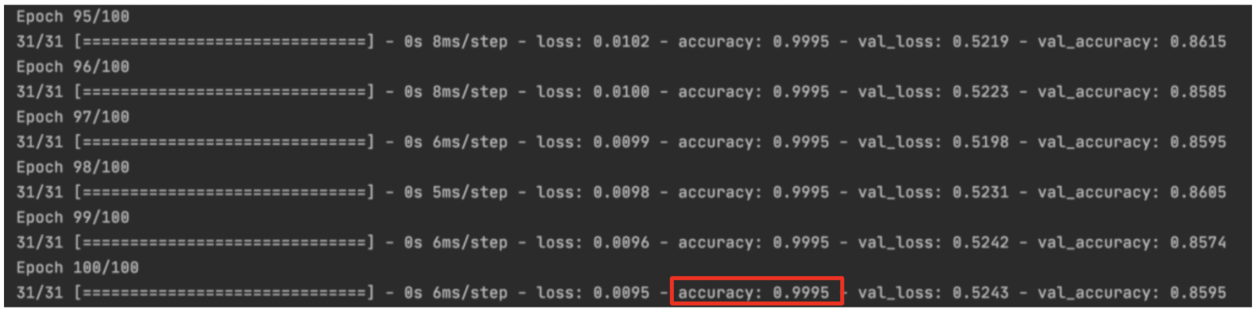
Other parameters that we experimented with were the number of hidden layers and the number of neurons in each layer. First, we thought that the more, the merrier and plugged in huge numbers and copy-pasted to create many layers. However, the testing accuracy was not increasing with the number of neurons or the number of layers. During the experimentation, we gathered that going from the high number of neurons to smaller and smaller resulted in high testing accuracy. Hence, we started with the number of pixels in an image (784) for the first layer. Consequently, we divided the number of neurons by two for all following layers. Stopping at 98 on the last hidden layer showed the best result.

Hence, our best_trained_model has the first layer with 784 neurons, the ReLu activation function, and the Random Uniform kernel initializer.

Respectively, our best model has three hidden layers. Two have the same activation function and kernel initializer. The last hidden layer uses the Tanh activation function and the Glorot Uniform kernel initializer. Since the model is for classification, the last layer of the model has to use the Softmax activation function and he_normal kernel initializer.

2. Model Performance & Confusion Matrix (A)

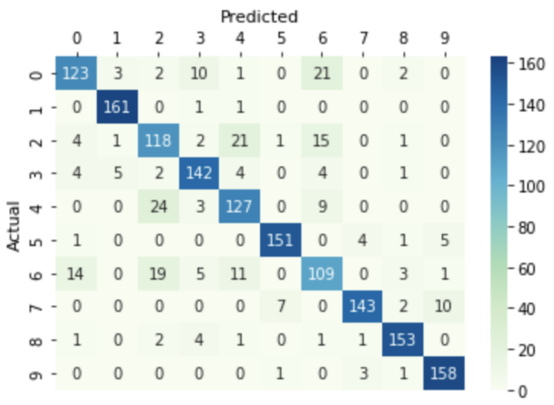
As mentioned in section 1 (Model & Training Procedure Description) our final model had a total of 5 layers. The model was then developed through 100 iterations using the validation set. We found that at the end of the development process. The validation set was being correctly predicted almost 100% of the time.



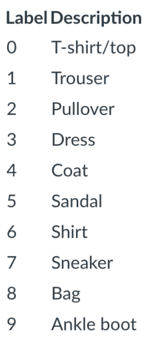
```
Epoch 95/100
31/31 [==============================] - 0s 8ms/step - loss: 0.0102 - accuracy: 0.9995 - val_loss: 0.5219 - val_accuracy: 0.8615
Epoch 96/100
31/31 [==============================] - 0s 8ms/step - loss: 0.0100 - accuracy: 0.9995 - val_loss: 0.5223 - val_accuracy: 0.8585
Epoch 97/100
31/31 [==============================] - 0s 6ms/step - loss: 0.0099 - accuracy: 0.9995 - val_loss: 0.5198 - val_accuracy: 0.8595
Epoch 98/100
31/31 [==============================] - 0s 5ms/step - loss: 0.0098 - accuracy: 0.9995 - val_loss: 0.5231 - val_accuracy: 0.8605
Epoch 99/100
31/31 [==============================] - 0s 6ms/step - loss: 0.0096 - accuracy: 0.9995 - val_loss: 0.5242 - val_accuracy: 0.8574
Epoch 100/100
31/31 [==============================] - 0s 6ms/step - loss: 0.0095 - accuracy: 0.9995 - val_loss: 0.5243 - val_accuracy: 0.8595
```

Image 1: Model Development

The final model was then evaluated on the training set and had an accuracy of 85.48%. Meaning that 1400 of the 1625 images in the training set had been correctly labeled. This performance can be visualized through the confusion matrix that our team plotted using the Seaborn library. This plot was carefully created using a colors scheme that could be easily interpreted and axis ranges that were selected within the context of the assignment.



	0	1	2	3	4	5	6	7	8	9
0	123	3	2	10	1	0	21	0	2	0
1	0	161	0	1	1	0	0	0	0	0
2	4	1	118	2	21	1	15	0	1	0
3	4	5	2	142	4	0	4	0	1	0
4	0	0	24	3	127	0	9	0	0	0
5	1	0	0	0	0	151	0	4	1	5
6	14	0	19	5	11	0	109	0	3	1
7	0	0	0	0	0	7	0	143	2	10
8	1	0	2	4	1	0	1	1	153	0
9	0	0	0	0	0	1	0	3	1	158



Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Image 2 & 3: Confusion Matrix & Labels

From the confusion matrix above it is clear that the Sequential model had little difficulty correctly labeling Trousers(1), Dresses(3), Sandals(5), Sneakers(7), Bags(8), and Ankle Boots(9).

The two challenges for the algorithms were discerning between Pullovers(2) & Coats(4), then Tshirt/top(0) & Shirt(6). This is presumably due to the similarities between how they look, which resulted in similarities during image processing and encoding. Even with just these four relatively challenging categories, our model had an accuracy rating of 74%.

3. Training Performance Plot(BAE)

Using our best performing ANN, the following figures were plotted to show how training accuracy and validation accuracy change over time during training. The x-axis represents the number of epochs and the y-axis represents the accuracy or loss.

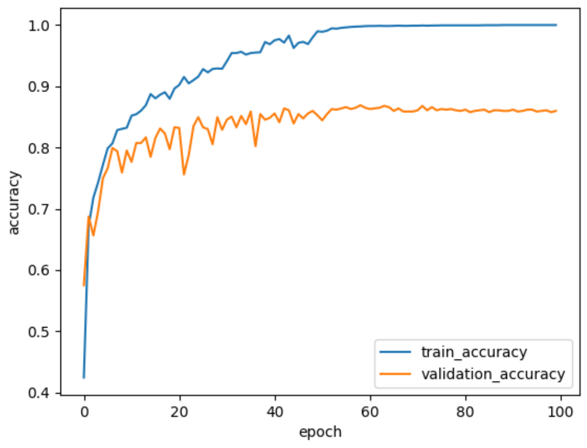


Image 4: Training and Validation accuracy / # of epochs

The training accuracy increased from 0.4 to 1 whereas the validation accuracy increased from 0.6 to around 0.85.

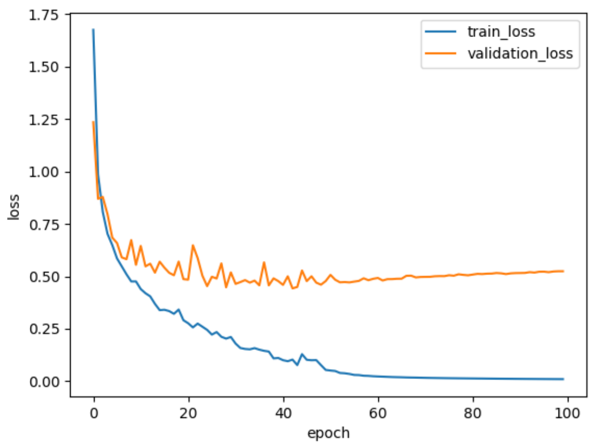


Image 5: Training and Validation loss / # of epochs

The training loss decreased from 1.75 to 0 whereas the validation loss decreased from 1.25 to around 0.55.

4. Visualization(E)

According to the error percentage we obtained from the model.evaluate() function, 14.52% of the images were misclassified. To understand what kind of images (label-wise) were misclassified by our best performing model, we compared the prediction and test values and plotted the images if not matched as shown below (this image can be found in the submission, titled “misclassified_images.png”).

Image 6: Misclassified Images

There were 236 out of 1625 test images that the model predicted incorrectly, which matches our error percentage $(236/1625) \ * \ 100 \ = \ 14.52\%$. Based on the images with the actual and prediction labels titled, we stored the frequency of the misclassification in a dictionary and sorted it in reverse order (most frequent to least frequent) to see which inaccurate predictions were the most commonly made. As portrayed in the confusion matrix above, the top three frequent mistakes made by our model were shown in the images below and also as follows: {'*0, 6*': 26, '*2, 4*': 23, '*4, 2*': 20} when the actual and predicted label values are in the keys respectively and the values are the frequencies of each key. The actual label was 0 (T-shirt/top) whereas our model predicted 6 (shirt) instead. When the actual label was 2 (pullover), the model predicted 4 (coat) and vice versa.

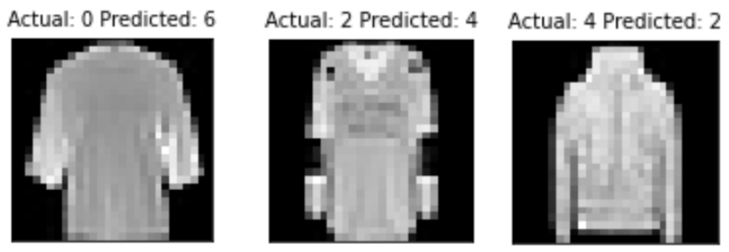


Image 7, 8 & 9: Example T-shirt, pullover, coat

As our group was going over the input images, we found out that the key difference between T-shirts and shirts was the buttons and the difference between pullovers and coats was whether they had a zipper or not. Based on our observations, the key factor of misclassification could be that the model was not able to perfectly distinguish the zipper and buttons. If there were to be clearer images or more data, the model would have been able to predict the labels more accurately.