Assignment 4

Charlotte Sacre - Theodore Glavas - Wassim Jabbour

I. Implementation

We started by processing the images into correctly sized arrays using help from this notebook. We then Implemented k-Fold cross validation on the dataset by splitting the entire training data into 5 folds to train 5 separate models, each with a different fold acting as validation. This was to take advantage of all the data available to us.. We then used a multilayer CNN model inspired from this notebook. The final model structure is as follows:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        320

 batch_normalization (BatchN  (None, 28, 28, 32)       128
 ormalization)

 conv2d_1 (Conv2D)           (None, 28, 28, 32)        9248

 batch_normalization_1 (Batc  (None, 28, 28, 32)       128
 hNormalization)

 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
 )

 dropout (Dropout)           (None, 14, 14, 32)        0

 conv2d_2 (Conv2D)           (None, 14, 14, 32)        9248

 batch_normalization_2 (Batc  (None, 14, 14, 32)       128
 hNormalization)

 dropout_1 (Dropout)         (None, 14, 14, 32)        0

 conv2d_3 (Conv2D)           (None, 14, 14, 64)        18496

 batch_normalization_3 (Batc  (None, 14, 14, 64)       256
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 64)         0
 2D)

 dropout_2 (Dropout)         (None, 7, 7, 64)          0

 flatten (Flatten)           (None, 3136)              0

 dense (Dense)               (None, 350)               1097950

 batch_normalization_4 (Batc  (None, 350)              1400
 hNormalization)

 dropout_3 (Dropout)         (None, 350)               0

 dense_1 (Dense)             (None, 100)               35100

 batch_normalization_5 (Batc  (None, 100)              400
 hNormalization)

 dense_2 (Dense)             (None, 10)                1010

=================================================================
Total params: 1,173,812
Trainable params: 1,172,592
Non-trainable params: 1,220
_________________________________________________________________
```

To improve on the results from this notebook, we tried the following parameter tuning:

- Set the number of filters in the 4 CNN layers to 32/32/32/64
- Added padding to each CNN layer
- Tried increasing or decreasing number of CNN layers, but 4 was the best
- Reduced the size of the Dense network to 350/100 for less overfitting
- Increased dropout layers to 0.3/0.3/0.3/0.6 for less overfitting
- Increased the number of training epochs to 250, with a stop condition at 90.05% validation accuracy
- Used Adam learning rate optimizer with parameter of 0.01
- Added K-Fold cross validation and trained 5 separate models

To predict test labels, the predictions from the 5 models were combined and the most frequent prediction was chosen for each label. This technique contributed about an 0.8% boost in test accuracy.

II. Results

With this model, we obtained a score of 0.9130. Training accuracy was at approximately 0.96, while validation accuracy hovered around 0.907 on average.

```
Score per fold
------------------------------------------------------------------------
> Fold 1 - Loss: 0.29989317059516907 - Accuracy: 91.07999801635742%
------------------------------------------------------------------------
> Fold 2 - Loss: 0.35691705346107483 - Accuracy: 91.0099983215332%
------------------------------------------------------------------------
> Fold 3 - Loss: 0.3783420920372009 - Accuracy: 90.39999842643738%
------------------------------------------------------------------------
> Fold 4 - Loss: 0.3359530568122864 - Accuracy: 91.00000262260437%
------------------------------------------------------------------------
> Fold 5 - Loss: 0.3854862451553345 - Accuracy: 90.21999835968018%
------------------------------------------------------------------------
Average scores for all folds:
> Accuracy: 90.74199914932251 (+- 0.35835231309123083)
> Loss: 0.35131832361221316
```

III. Challenges

Data preprocessing was difficult to get started, but we received guidance from a Kaggle notebook mentioned above to load in the dataset. We found it hard to find the right configuration of layers. Furthermore, we had an issue where we forgot to use argmax to return the label instead of random float values. We also forgot to download some dependencies.

When optimising the model, there was a large problem with overfitting. Validation accuracy always plateaued around 90%, while training accuracy quickly reached

more than 99%. Dropout, reduced dense network size and learning rate tuning helped raise that ceiling by a few tenths of a percent.

IV. Conclusion

We learned how to manipulate CNNs and work with them using tensorflow. Furthermore, we learned that even with a very simple CNN architecture, we are able to classify our clothing items pretty accurately. However, more complex network can perform better, but every additional gain in accuracy requires exponentially more work.

V. Individual Contribution

It was truly a team effort, we sat down together and brainstormed how to approach the problem. Theodore focused on optimising the model code, while Charlotte and Wassim did the predicting and exporting to csv.