Fashion MNIST Report

Problem Statement

Fashion-MNIST is a small dataset for fashion product classification consisting of 60,000 training images and 10,000 test images. All images are grayscale 28x28 pixels and there are 10 classes of clothing.
The following document discusses ideas and presents some models for the classification of these images.

Idea and approach

The base idea is that this dataset is not much more complex than the traditional MNIST dataset. With that assumption, we will try using some network architectures that are known to perform well on the MNIST dataset. Also, I tried training the model with just one part of the dataset for faster training (10,000 train images) and also with the whole dataset.

I tried running the models without data augmentation and the results were much worse. When doing data augmentation I used zooming, rotation, shifting, and shear.

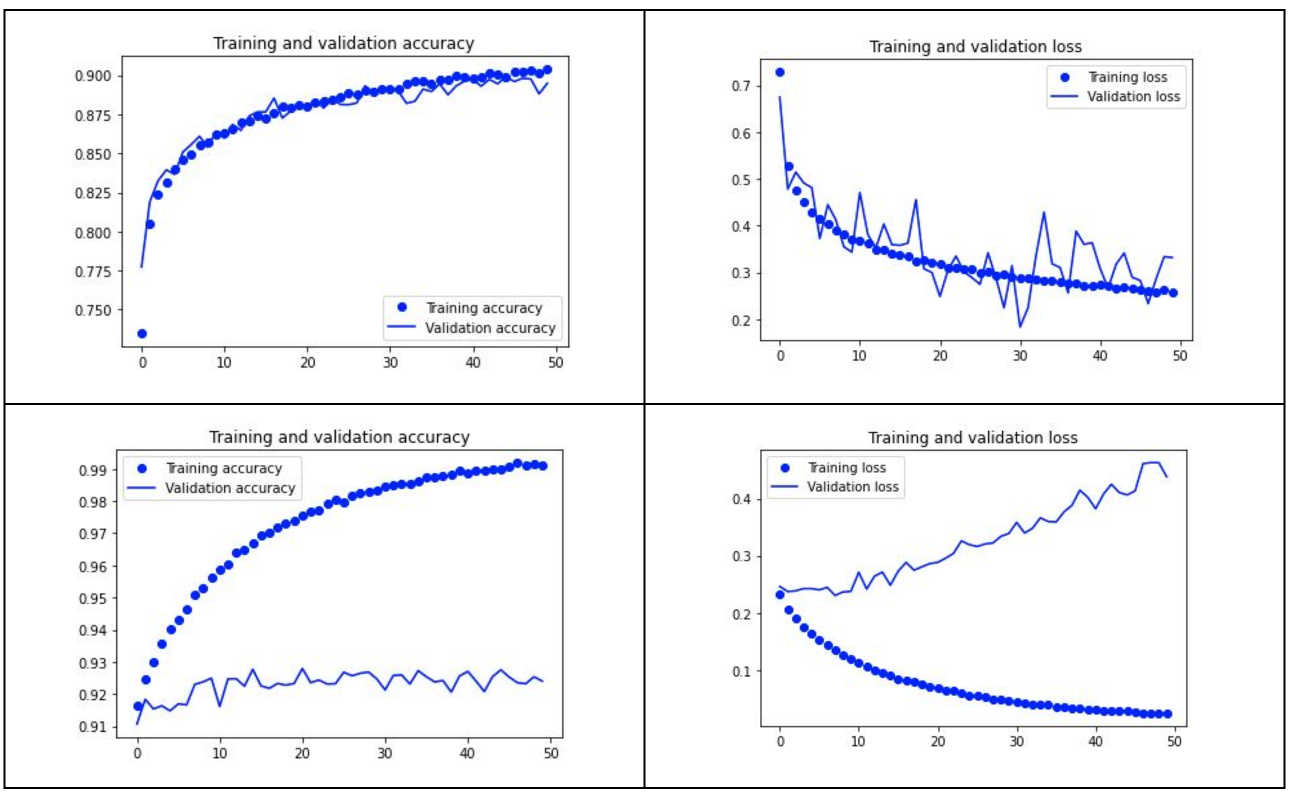
As for the loss function, all models use Categorical-Crossentropy

1 Layer CNN

The first approach is a simple neural network with one convolutional layer that has 693, 932 trainable parameters. The following image represents the network architecture.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 13, 13, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 5408)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               692352
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 693,962
Trainable params: 693,962
Non-trainable params: 0
```

This architecture works surprisingly well given its simplicity. Here, we have 91.39% accuracy on the validation dataset. Training this network on Google Colab with GPU acceleration took ~15min.

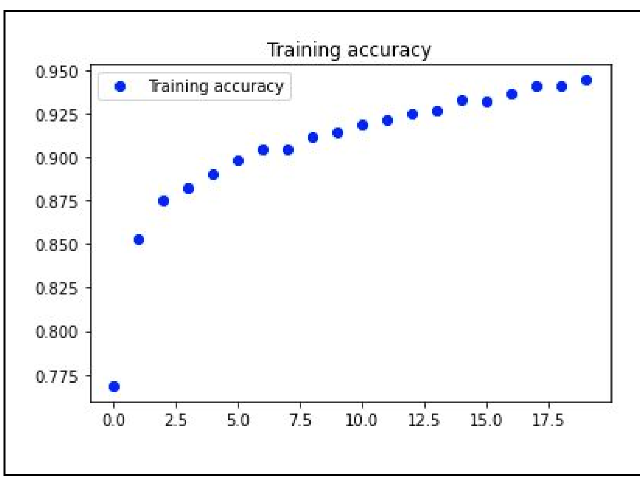


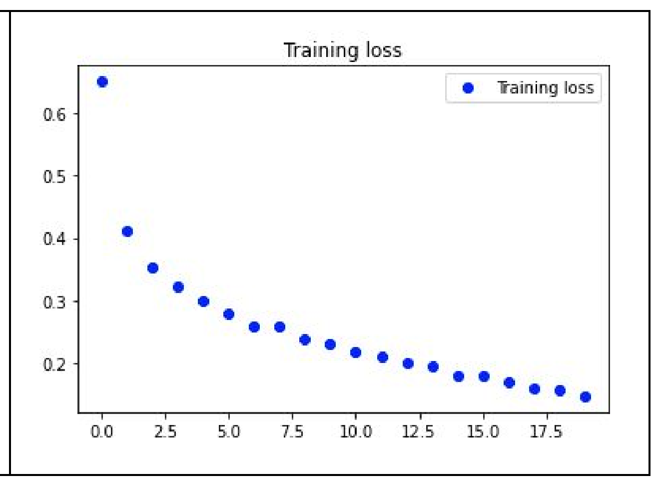
LeNet-5

For the second approach, I chose LeNet-5, an old CNN architecture proven to be reliable for simple image classification tasks, especially on small resolution images. Here we have 136,586 parameters and training doesn't take more than 15 minutes.

With LeNet-5 the training accuracy is 95.28% and validation accuracy is 92.11%.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_10 (Conv2D)           (None, 32, 32, 6)         156
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 16, 16, 6)         0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 16, 16, 16)        2416
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 8, 8, 16)          0
_________________________________________________________________
flatten_6 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_16 (Dense)             (None, 120)               123000
_________________________________________________________________
dense_17 (Dense)             (None, 84)                10164
_________________________________________________________________
dense_18 (Dense)             (None, 10)                850
=================================================================
Total params: 136,586
Trainable params: 136,586
Non-trainable params: 0
```



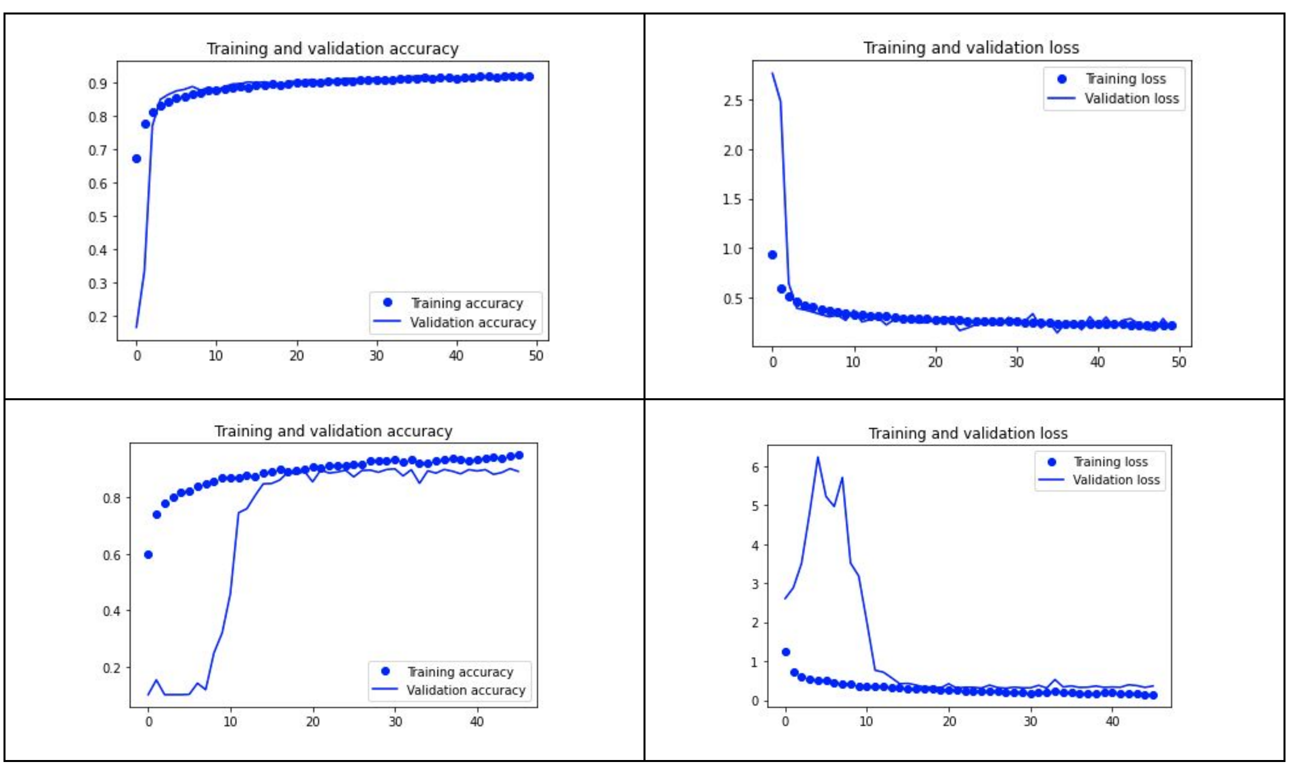


4 Layer CNN

The second approach was simply to stack more layers. Here we have a neural network with 4 convolutional layers that has 1,221,546 parameters. After adding padding we get 3 times more parameters. This model gave the best results, 92.51%.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
batch_normalization_7 (Batch (None, 26, 26, 32)        128
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 24, 24, 32)        9248
_________________________________________________________________
batch_normalization_8 (Batch (None, 24, 24, 32)        128
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 12, 12, 32)        0
_________________________________________________________________
dropout_6 (Dropout)          (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 10, 10, 64)        18496
_________________________________________________________________
batch_normalization_9 (Batch (None, 10, 10, 64)        256
_________________________________________________________________
dropout_7 (Dropout)          (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
batch_normalization_10 (Batc (None, 8, 8, 128)         512
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout_8 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               1049088
_________________________________________________________________
batch_normalization_11 (Batc (None, 512)               2048
_________________________________________________________________
dropout_9 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               65664
_________________________________________________________________
batch_normalization_12 (Batc (None, 128)               512
_________________________________________________________________
dropout_10 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_6 (Dense)              (None, 10)                1290
=================================================================
Total params: 1,221,546
Trainable params: 1,219,754
Non-trainable params: 1,792
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_15 (Conv2D)           (None, 28, 28, 32)        320
_________________________________________________________________
batch_normalization_12 (Batc (None, 28, 28, 32)        128
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 28, 28, 32)        9248
_________________________________________________________________
batch_normalization_13 (Batc (None, 28, 28, 32)        128
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
dropout_11 (Dropout)         (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 14, 14, 64)        18496
_________________________________________________________________
batch_normalization_14 (Batc (None, 14, 14, 64)        256
_________________________________________________________________
dropout_12 (Dropout)         (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 14, 14, 128)       73856
_________________________________________________________________
batch_normalization_15 (Batc (None, 14, 14, 128)       512
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
dropout_13 (Dropout)         (None, 7, 7, 128)         0
_________________________________________________________________
flatten_6 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_17 (Dense)             (None, 512)               3211776
_________________________________________________________________
batch_normalization_16 (Batc (None, 512)               2048
_________________________________________________________________
dropout_14 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_18 (Dense)             (None, 128)               65664
_________________________________________________________________
batch_normalization_17 (Batc (None, 128)               512
_________________________________________________________________
dropout_15 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_19 (Dense)             (None, 10)                1290
=================================================================
Total params: 3,384,234
Trainable params: 3,382,442
Non-trainable params: 1,792
```



Results

In the following table, you can see the results for different model architectures, data preparation, hyperparameters, and more.
Validation is done on 10,000 images from the Fashion-MNIST dataset which are normalized and augmented the same way as the training images.

	1 Layer CNN	**1 Layer CNN**	**4 Layer CNN**	**4 Layer CNN**	**LeNet-5**
Data	1 fold	Full dataset	1 fold	Full dataset	Full dataset
Augmentation	Normalization	Normalization, rotation, zoom, shear, shift	Normalization	Normalization, rotation, zoom, shear, shift	Normalization
Loss function	Categorical Crossentropy	Categorical Crossentropy	Categorical Crossentropy	Categorical Crossentropy	Sparse Categorical Crossentropy
# of parameters	693,932	693,932	1,219,754	1,219,754	136,586
Hyperparameters	Batch size: 256 Epochs: 100	Batch size: 256 Epochs: 50	Batch size: 256 Epochs: 100	Batch size: 256 Epochs: 50	Batch size: 32 Epochs: 20
Training accuracy	99.13%	90.30%	95.23%	91.19%	95.28%
Validation accuracy	91.97%	90.29%	89.92%	92.51%	92.11%

Future improvements

- RMSprop and Adam for each model
- Data augmentation methods
  - Rescale
  - Horizontal flip
  - Vertical flip
- Different batch sizes and epochs