Fashion Image Classification with Deep Learning Architectures

DEEP LEARNING - PROJECT 1

SOFIA BALTZI

Contents

Introduction

The purpose of this document is to report the results for the 1st Assignment of class Deep Learning, February – April 2021. The objective is to create a classifier, using deep learning architectures, that solves the problem of multiclass image classification.

The dataset used is the Fashion-MNIST. It consists of a training set of 60,000 images along with their labels and a test set of 10,000 images along with their labels as well. Images are grayscale with dimensions 28×28 and they belong to one of 10 classes. In both sets, classes are well balanced. Below there can be seen ten image examples, one per class.

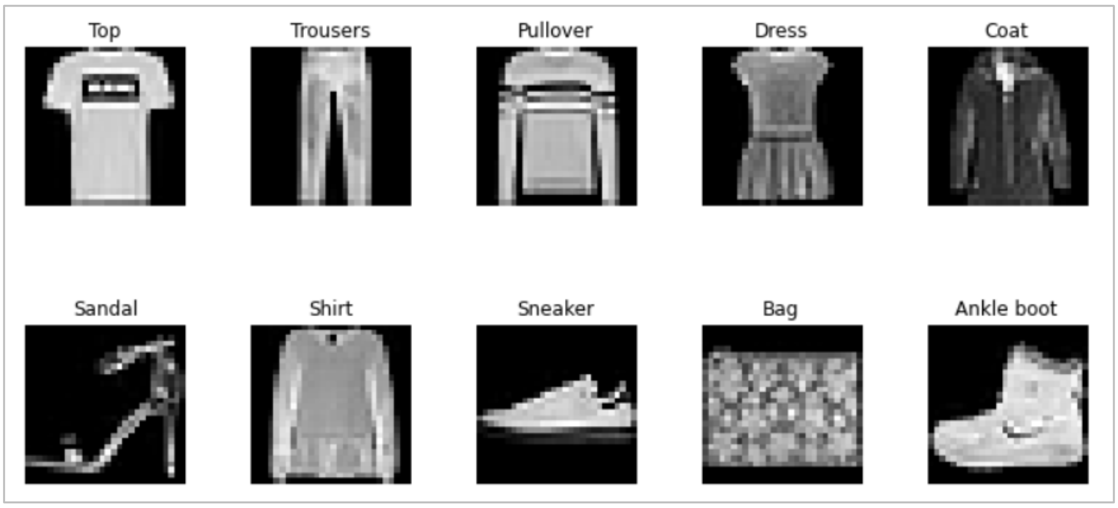


Figure 1: Image examples with labels

**Code for the project can be found here.*

Data Preprocessing

Images are represented by 28×28 arrays. Their values are divided by 255 – they are standardized – as a preprocessing step. The array of labels is transformed to a 1-hot array, each row representing one example and each column representing one class.

Finally, training set consists of 60,000 examples of size [28,28] and their corresponding labels in an array of shape [60,000, 10]. Test set consists of 10,000 examples of size [28,28] and their corresponding labels in an array of shape [10,000, 10].

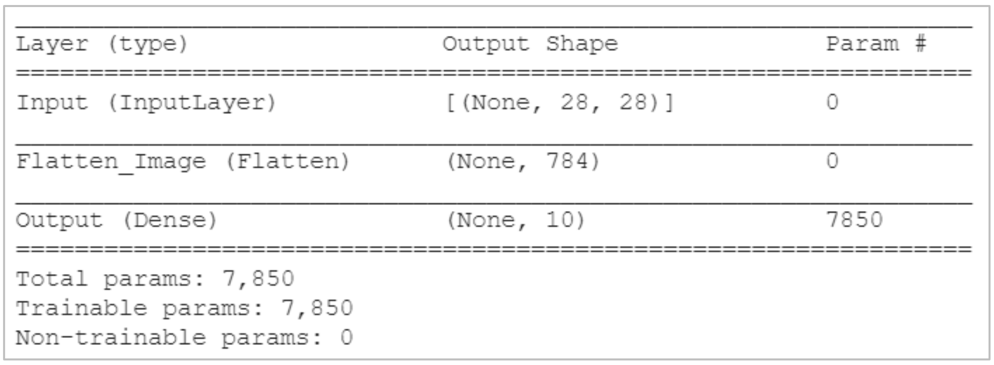
All models are implemented using Tensorflow/ Keras library.

Architecture 1 – MLP

For the implementation of the MLP models, the problem is treated as numeric. The images are inserted in the network as [28,28] arrays and then are flattened using the Flatten() layer to arrays of shape [,784].

Baseline

As a first attempt to solve the problem, a very simple model is implemented, consisting of an input layer, a flatten layer and a dense layer as the output.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input (InputLayer)           [(None, 28, 28)]          0
_________________________________________________________________
Flatten_Image (Flatten)      (None, 784)               0
_________________________________________________________________
Output (Dense)               (None, 10)                7850
=================================================================
Total params: 7,850
Trainable params: 7,850
Non-trainable params: 0
```

Figure 2: Baseline model

The model is trained in batches of 128 for 100 epochs, monitoring validation accuracy in a split equal to 20% of the training set. Adam optimizer is used to minimize the loss function that is categorical crossentropy. Early stopping is used as a callback, terminating the training process if a better validation accuracy is not achieved for 10 consecutive epochs. The output layer uses Softmax as activation function, to produce probabilities for the 10 classes.

Training process terminates after 44 epochs giving 87.07% accuracy on the training set and 84.60% accuracy on the test set. There is still learning capability, since the model performs relatively poorly on the training set.

```
Train accuracy: 0.8707
Test accuracy: 0.846
```

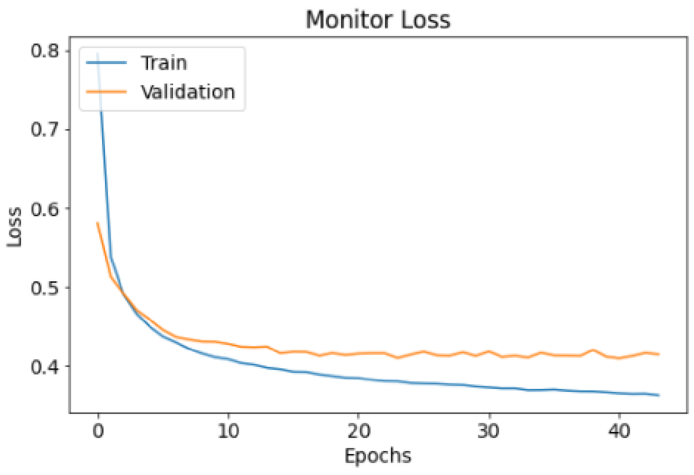
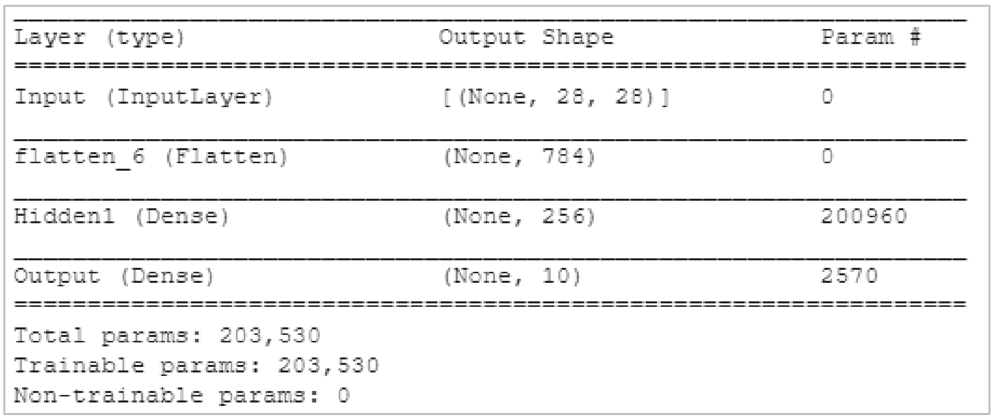


Figure 3: Baseline Results

Baseline + 1 Hidden Layer

For the second attempt, the same architecture as the baseline is used, with the addition of one hidden layer.

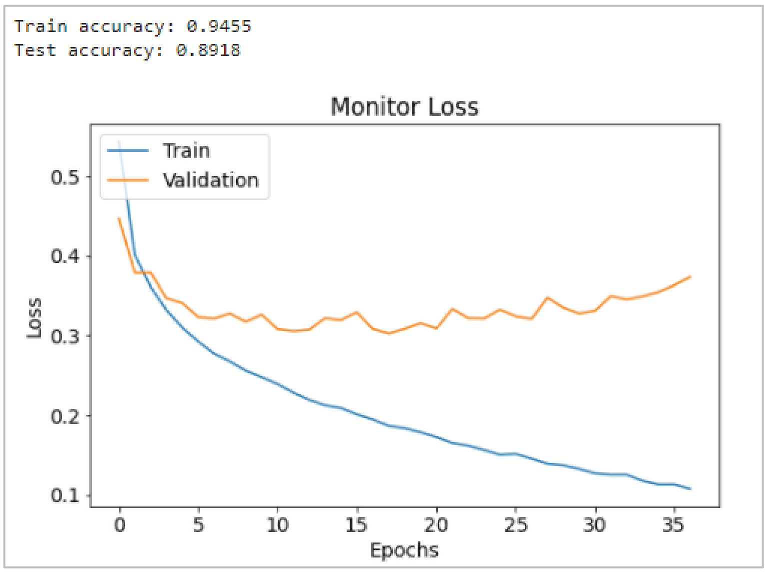


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input (InputLayer)           [(None, 28, 28)]          0
_________________________________________________________________
flatten_6 (Flatten)          (None, 784)               0
_________________________________________________________________
Hidden1 (Dense)              (None, 256)               200960
_________________________________________________________________
Output (Dense)               (None, 10)                2570
=================================================================
Total params: 203,530
Trainable params: 203,530
Non-trainable params: 0
```

Figure 4: Baseline + 1 Hidden layer model

The model is trained in batches of 128 for 100 epochs, monitoring validation accuracy in a split equal to 20% of the training set. Adam optimizer is used to minimize the loss function that is categorical crossentropy. Early stopping is used as a callback, terminating the training process if a better validation accuracy is not achieved for 10 consecutive epochs. The hidden layer uses ReLU as activation function and the output layer uses Softmax, to produce probabilities for the 10 classes.

Training process terminates after 37 epochs giving 94.55% accuracy on the training set and 89.18% accuracy on the test set. The model managed to learn the training set very well but there was not much improvement in the test set score, leading to the conclusion that it is probably overfitting. The case of overfitting can also be observed on the diagram, where training loss continuously drops whereas validation loss starts to increase after the 30th epoch.

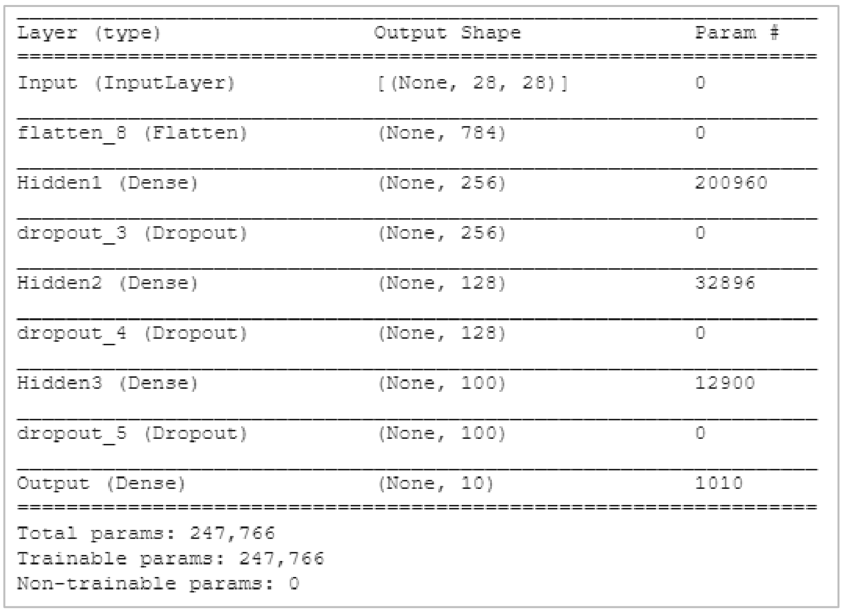


```
Train accuracy: 0.9455
Test accuracy: 0.8918
```

Figure 5: Baseline + 1 Hidden layer Results

Baseline + 3 Hidden Layers + Dropout

For the third attempt, there are used three hidden layers with dropout 0.4 after each one, to avoid overfitting.

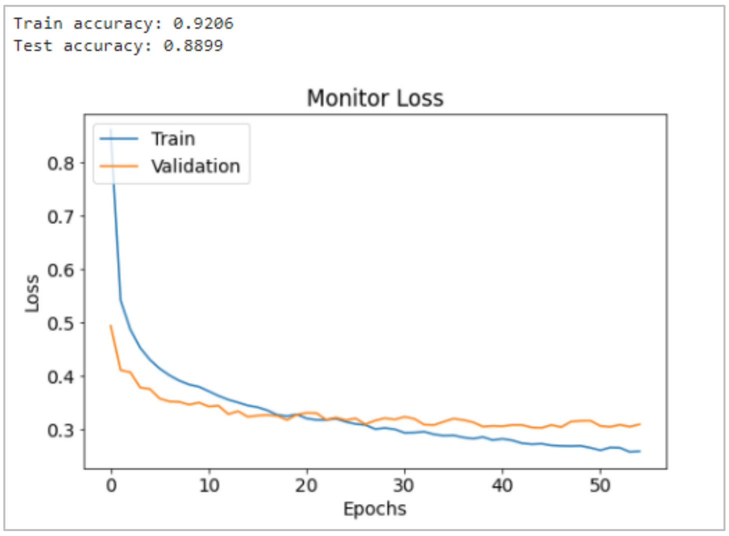


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
Input (InputLayer)           [(None, 28, 28)]          0         
_________________________________________________________________
flatten_8 (Flatten)          (None, 784)               0         
_________________________________________________________________
Hidden1 (Dense)              (None, 256)               200960    
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0         
_________________________________________________________________
Hidden2 (Dense)              (None, 128)               32896     
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0         
_________________________________________________________________
Hidden3 (Dense)              (None, 100)               12900     
_________________________________________________________________
dropout_5 (Dropout)          (None, 100)               0         
_________________________________________________________________
Output (Dense)               (None, 10)                1010      
=================================================================
Total params: 247,766
Trainable params: 247,766
Non-trainable params: 0
```

Figure 6: Baseline + 3 Hidden layers + Dropout model

The model is trained in batches of 128 for 100 epochs, monitoring validation accuracy in a split equal to 20% of the training set. Adam optimizer is used to minimize the loss function that is categorical crossentropy. Early stopping is used as a callback, terminating the training process if a better validation accuracy is not achieved for 10 consecutive epochs. The hidden layers use ReLU as activation function and the output layer uses Softmax, to produce probabilities for the 10 classes.

Training process terminates after 55 epochs giving 92.06% accuracy on the training set and 88.99% accuracy on the test set. Training accuracy has dropped, meaning that there is not much of an overfitting problem, but the test accuracy is still low - even lower than with the simpler model. To attempt increasing the accuracy of our solution we attempt a different architecture.



```
Train accuracy: 0.9206
Test accuracy: 0.8899
```

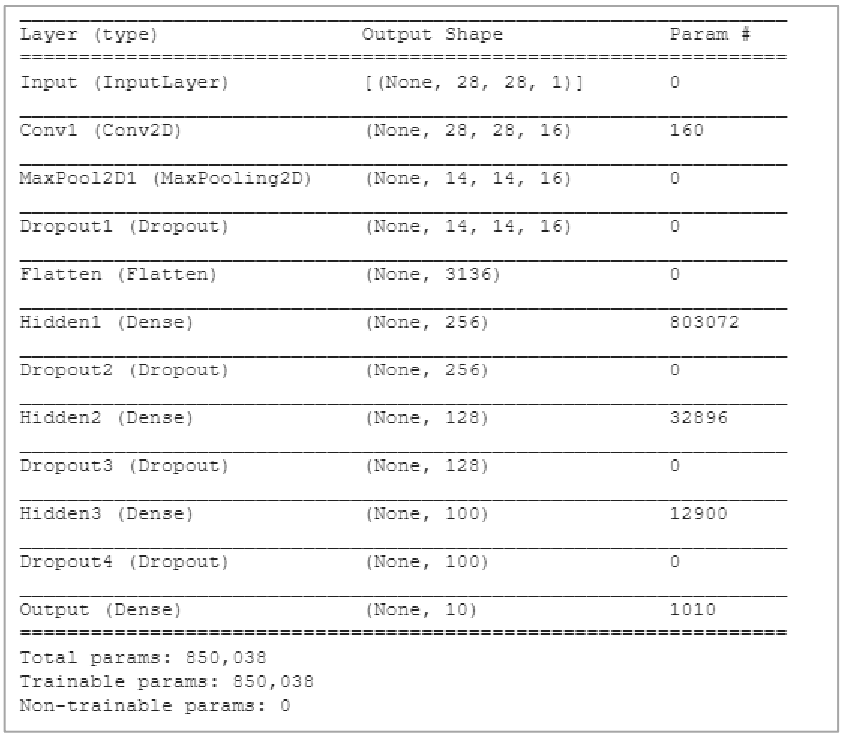
Figure 7: Baseline + 3 Hidden layers + Dropout Results

Architecture 2 – CNN

Presented in this section, are the experiments in which CNN architectures were used. Instead of a flat image, networks get as input arrays of size [28,28,1] (images of size 28x28 in grayscale).

1 CNN Layer + Best MLP

The best accuracy achieved with just MLP layers was 88.99% on the test set. For this experiment, we keep that architecture (Baseline + 3 Hidden Layers + Dropout) but add one convolutional layer as the first layer, with 16 (3x3) kernels and ReLU as activation function, followed by (2x2) MaxPooling with stride 2.

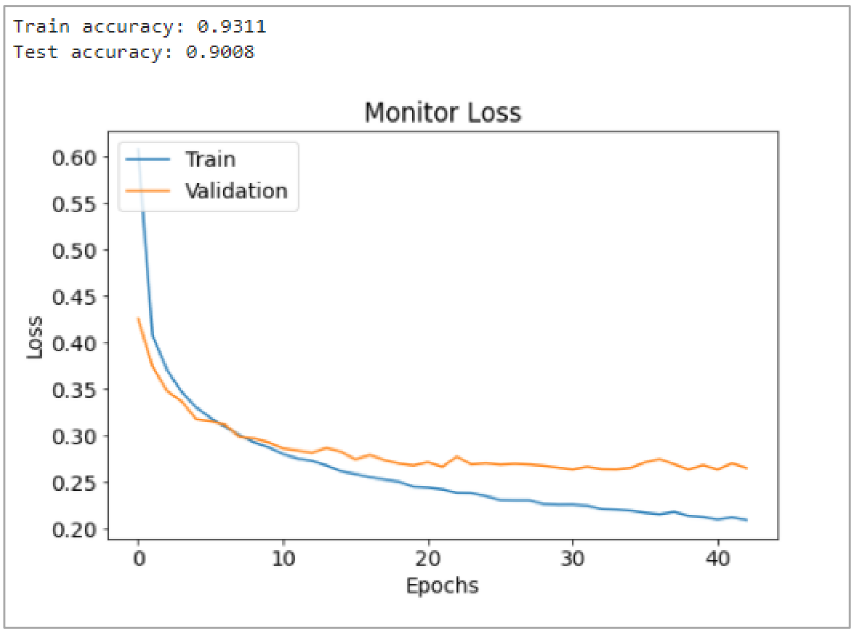


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input (InputLayer)           [(None, 28, 28, 1)]       0
_________________________________________________________________
Conv1 (Conv2D)               (None, 28, 28, 16)        160
_________________________________________________________________
MaxPool2D1 (MaxPooling2D)    (None, 14, 14, 16)        0
_________________________________________________________________
Dropout1 (Dropout)           (None, 14, 14, 16)        0
_________________________________________________________________
Flatten (Flatten)            (None, 3136)              0
_________________________________________________________________
Hidden1 (Dense)              (None, 256)               803072
_________________________________________________________________
Dropout2 (Dropout)           (None, 256)               0
_________________________________________________________________
Hidden2 (Dense)              (None, 128)               32896
_________________________________________________________________
Dropout3 (Dropout)           (None, 128)               0
_________________________________________________________________
Hidden3 (Dense)              (None, 100)               12900
_________________________________________________________________
Dropout4 (Dropout)           (None, 100)               0
_________________________________________________________________
Output (Dense)               (None, 10)                1010
=================================================================
Total params: 850,038
Trainable params: 850,038
Non-trainable params: 0
```

Figure 8: 1 CNN layer + Best MLP model

The model is trained in batches of 128 for 100 epochs, monitoring validation accuracy in a split equal to 20% of the training set. Adam optimizer is used to minimize the loss function that is categorical crossentropy. Early stopping is used as a callback, terminating the training process if a better validation accuracy is not achieved for 10 consecutive epochs. The hidden and convolutional layers use ReLU as activation function and the output layer uses Softmax, to produce probabilities for the 10 classes.

Training process terminates after 44 epochs giving 93.11% accuracy on the training set and 90.08% accuracy on the test set. We managed to improve test accuracy – about 1% - while keeping training accuracy relatively low. For the next experiment, we attempt to remove some of the dense layers and add convolutional layers, since this seems to work.

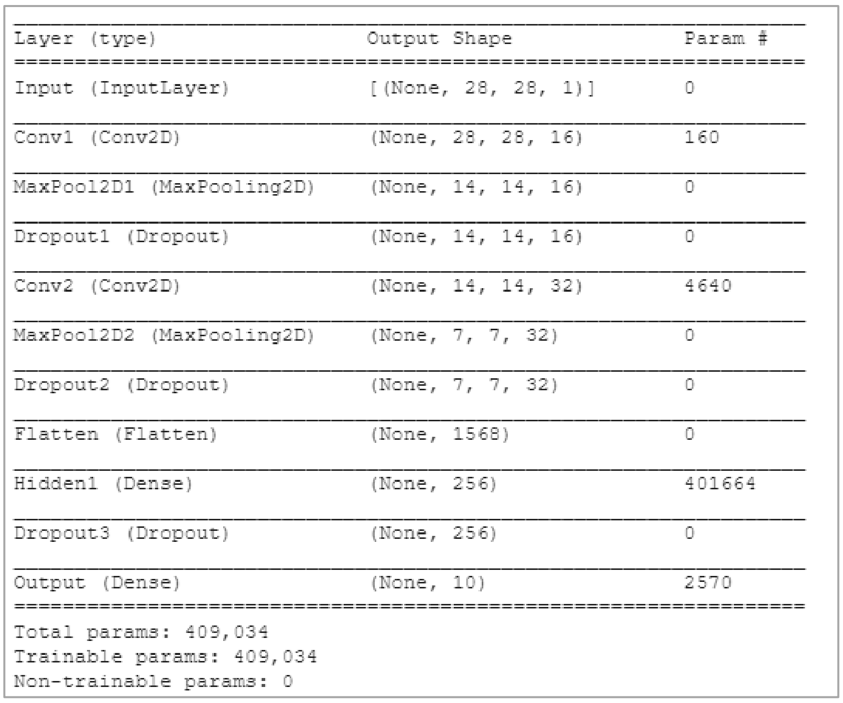


```
Train accuracy: 0.9311
Test accuracy: 0.9008
```

Figure 9: 1 CNN layer + Best MLP Results

2 CNN Layers + 1 Hidden Layer

For this experiment, the first layer is a convolutional layer, with 16 (3x3) kernels and ReLU as activation function, followed by (2x2) MaxPooling with stride 2. Then, follows another convolutional layer, with 32 (3x3) kernels and ReLU as activation function, followed by (2x2) MaxPooling with stride 2. The output is flattened and passes through a dense layer and then the output layer.

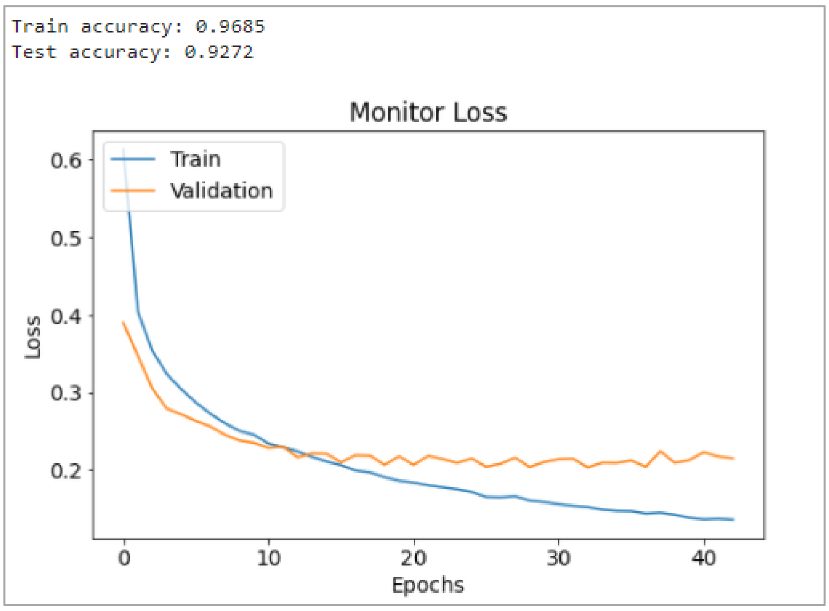


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input (InputLayer)           [(None, 28, 28, 1)]       0
_________________________________________________________________
Conv1 (Conv2D)               (None, 28, 28, 16)        160
_________________________________________________________________
MaxPool2D1 (MaxPooling2D)    (None, 14, 14, 16)        0
_________________________________________________________________
Dropout1 (Dropout)           (None, 14, 14, 16)        0
_________________________________________________________________
Conv2 (Conv2D)               (None, 14, 14, 32)        4640
_________________________________________________________________
MaxPool2D2 (MaxPooling2D)    (None, 7, 7, 32)          0
_________________________________________________________________
Dropout2 (Dropout)           (None, 7, 7, 32)          0
_________________________________________________________________
Flatten (Flatten)            (None, 1568)              0
_________________________________________________________________
Hidden1 (Dense)              (None, 256)               401664
_________________________________________________________________
Dropout3 (Dropout)           (None, 256)               0
_________________________________________________________________
Output (Dense)               (None, 10)                2570
=================================================================
Total params: 409,034
Trainable params: 409,034
Non-trainable params: 0
```

Figure 9: 2 CNN layers + 1 Hidden layer model

The model is trained in batches of 128 for 100 epochs, monitoring validation accuracy in a split equal to 10% of the training set. Adam optimizer is used to minimize the loss function that is categorical crossentropy. Early stopping is used as a callback, terminating the training process if a better validation accuracy is not achieved for 10 consecutive epochs. The hidden and convolutional layers use ReLU as activation function and the output layer uses Softmax, to produce probabilities for the 10 classes.

Training process terminates after 43 epochs giving 96.85% accuracy on the training set and 92.72% accuracy on the test set. That is an improvement of about 3.7 compared to our best MLP model and 2.6 compared to the other CNN model.



```
Train accuracy: 0.9685
Test accuracy: 0.9272
```

Figure 10: 2 CNN layers + 1 Hidden layer Results

Discussion

Fashion-MNIST image classification problem seems to be rather easy, since even the simplest architectures can give satisfying results. Nonetheless, architectures that use only dense layers seem to not be able to score more than 88 – 89 % on the test set. CNNs can match and outperform these networks by taking into account spatial information and using it to make decisions, leading to better results – accuracy over 92 %.

	Model	Test Accuracy (%)
MLPs	Baseline	84.60 %
	Baseline + 1 Hidden Layer	89.18 %
	Baseline + 3 Hidden Layers + Dropout	88.99 %
CNNs	1 CNN Layer + Best MLP	90.08 %
	2 CNN Layers + 1 Hidden Layer	92.72 %

Table 1: Summary of results