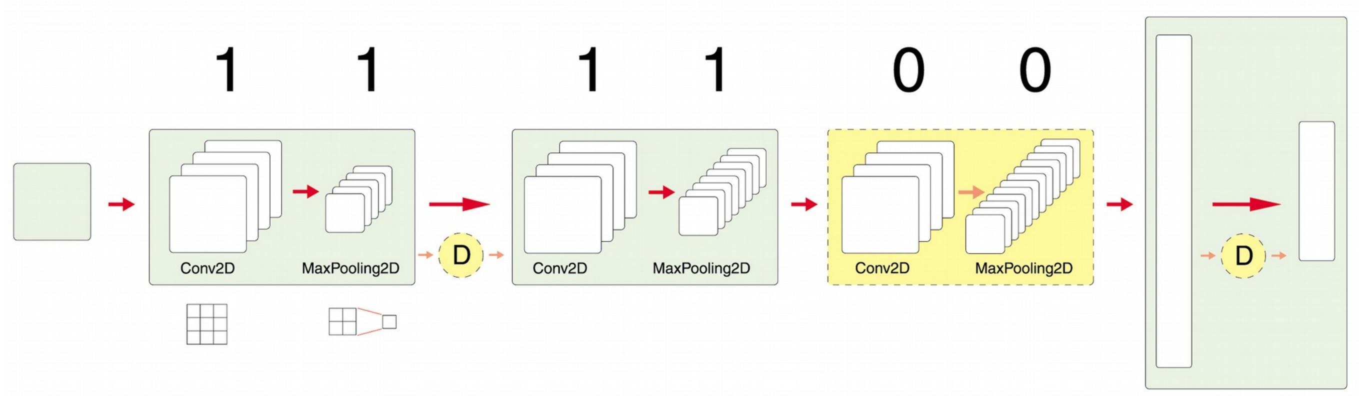
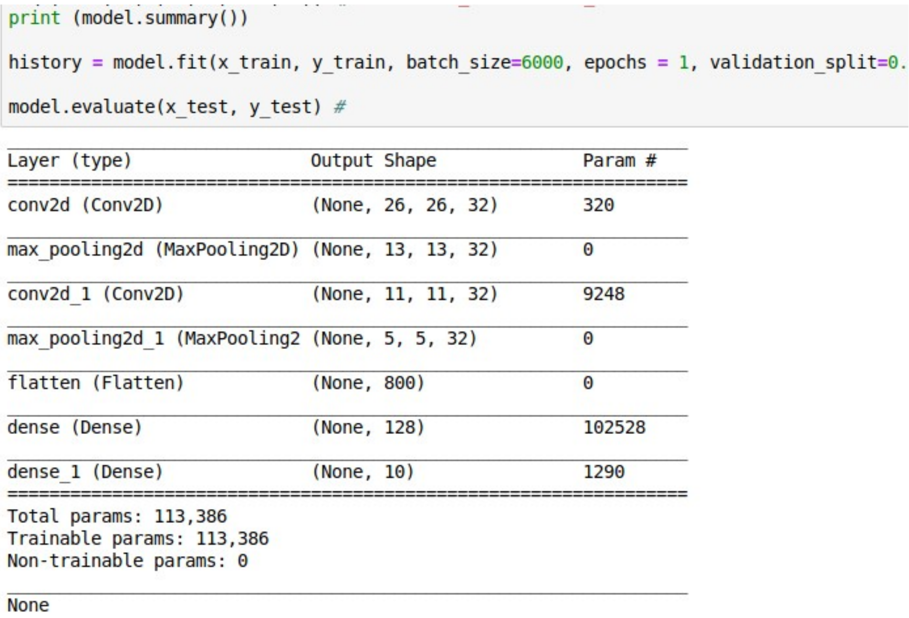
Report

Something about Convolutional Neural Networks



In this task, we were provided with a Convolutional Neural Networks previously created using the Tensorflow and Keras library.

Neural Network information is available on request and it is:



```
print (model.summary())

history = model.fit(x_train, y_train, batch_size=6000, epochs = 1, validation_split=0.

model.evaluate(x_test, y_test) #
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320

max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0

conv2d_1 (Conv2D)            (None, 11, 11, 32)        9248

max_pooling2d_1 (MaxPooling2 (None, 5, 5, 32)          0

flatten (Flatten)            (None, 800)               0

dense (Dense)                (None, 128)               102528

dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 113,386
Trainable params: 113,386
Non-trainable params: 0

None
```

This configuration will be considered basic and we will call the **main model.**

Changing the input parameters to investigate the work of the neural network in terms of computation time, loss function and accuracy.

Datasets to research the model: `tf.keras.datasets.fashion_mnist`

Experiment 1.

The first experiment was to run the program on various computing platforms.

tab 1

	Time(s, ms)	**Loss**	**Accuracy (%)**
LapTop	1583/5	0,415	84,90%
Colab	257/0,9	0,42	84,40%
Colab+GPU	33/0,112	0,41	85,00%

Conclusion 1.

As can be seen from the table, the execution speed in Colab + GPU is fundamentally higher. In the future, all experiments will be carried out on **Colab + GPU**

Experiment 2.

Differences in the values of loss function and accuracy motivated to conduct one simple experiment. Several runs of the model without changing the input parameters.

tab 2

	Time(s, ms)	Loss	Accuracy (%)
Main model	33/0,112	0,410	86,02%
Main model	--	0,439	84,60%
Main model	--	0,428	84,10%
Main model	--	0,409	85,14%
Main model	--	0,422	84,34%
arithmetic mean		0,418	84,84%

Conclusion 2

Output parameters are different, but not fundamentally and randomly. For more stable values of all experiments, it seems reasonable to conduct a series of model launches (for example, 10 each) and take into account the arithmetic mean. Using “shuffle=False” in fit-function gives less difference in results also.

Experiment 3.1

Here we will change the number of feature maps.

Recall in the base model 32 of feature maps.

tab 3.1

Feature maps	Time(s, ms)	Loss	Accuracy (%)
32 (repeat)	33/0,112	0,410	85,00%
16	--	0,422	84,90%
64	--	0,421	85,00%

Conclusion 3.1

Let's try other values:

Experiment 3.2

What is the minimum number of features maps with an accuracy of at least 0.8 ?

tab 3.2

Feature maps	Time(s, ms)	Loss	Accuracy (%)
32 (repeat)	33/0,112	0,410	85,00%
8	/ 0,09	0,450	83,40%
4	--	0,468	82,80%
3	--	0,537	79,99%
2	--	0,558	78,00%
1	--	0,584	78,60%

Experiment 4 (series similar experiments):

tab 4.1

Feature maps	Time(s)	Loss	Accuracy (%)
Main model **1-1-1-1**	33	0,410	85,00%
Remove the final Conv & Maxpool **1-1-0-0**	32	0,388	86,18%
Remove the final Convolution **1-1-0-1**	31	0,418	84,97%
Remove the final Maxpool **1-1-1-0**	35	0,362	87,15%
Remove all Conv & Maxpool **0-0-0-0**	26	0,448	84,30%
Remove all Convolution **0-1-0-1**	25	0,726	72,71%
Remove all Maxpool **1-0-1-0**	36	0,329	88,14%
Remove the first Convolution **0-1-1-1**	31	0,488	82,45%
Remove the first Maxpool **1-0-1-1**	37	0,328	88,35%
adding Convolutions **1-0-1-1 + 1**	36	0,337	87,79%
adding Conv & maxpool **1-0-1-1 + 1-1**	41	0,389	86,59%

Add Dropout layer

Tab 4.2

Between: 1-0 - D - 1-1	Loss	Accuracy (%)
Dropout 0,25	0,388	86,32%
Dropout 0,5	0,411	85,66%

Tab 4.3

Between full connection: 1-0-1-1_1-D-1	Loss	Accuracy (%)
Dropout 0,25	0,385	86,27%
Dropout 0,5	0,382	86,02%

Experiment 5:

How many epochs are optimal for maximum accuracy in TEST? (main model 1-1-1-1)

tab 5

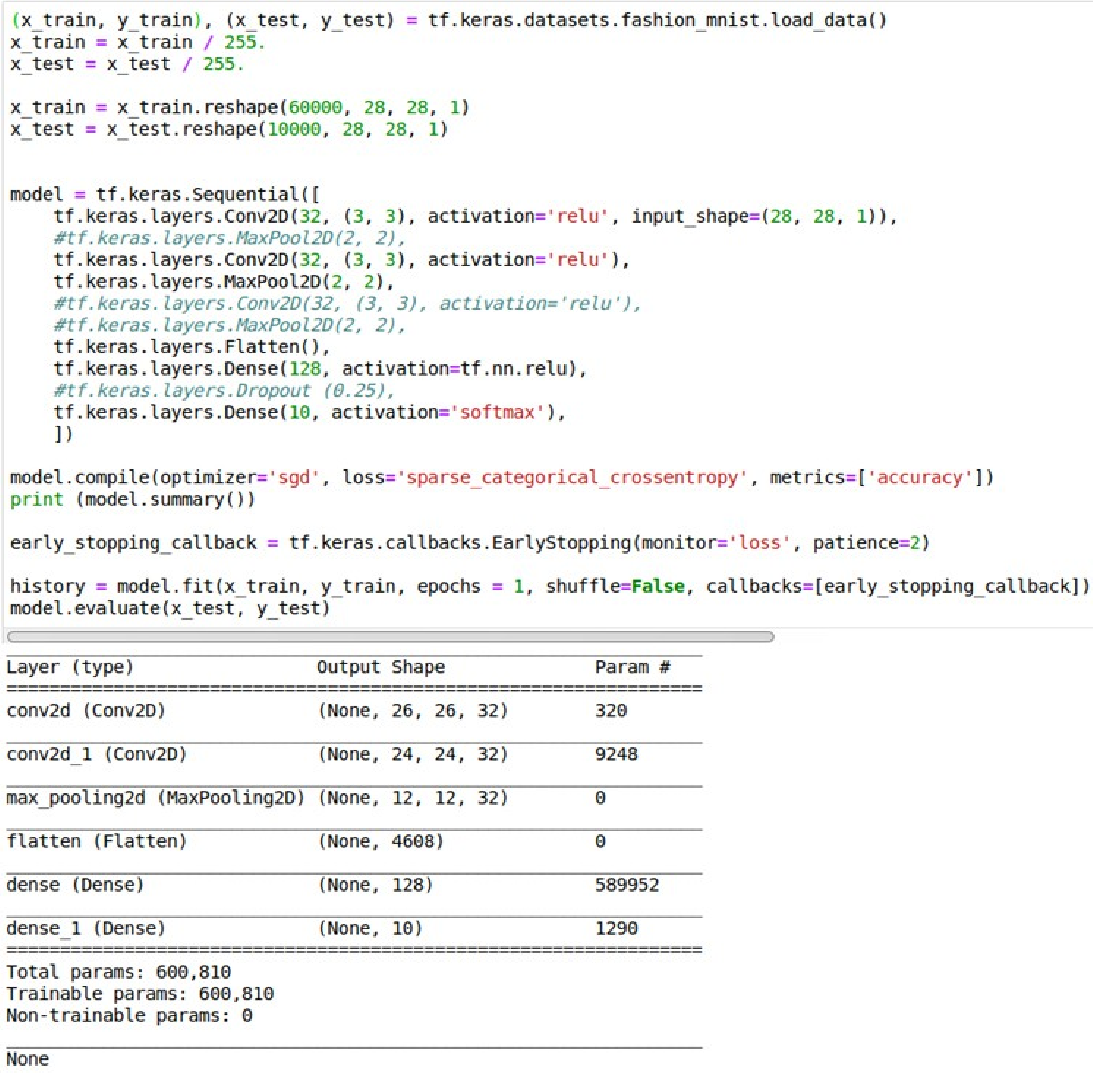
epochs	Time(s, ms)	Loss	Accuracy (%)
5	33 / 0,1	0,3100	86,70%
10	61 / 0,1	0,2965	89,46%
20	120 / 0,1	0,2790	90,33%
25	150 / 0,1	0,2770	90,28%
30	180 / 0,1	0,2850	90,59%
50	300 / 0,1	0,3470	90,25%
100	610 / 0,1	0,4032	90,72%

Finally Experiment

feature maps = 32, epochs=25, model 1-0-1-1

tab 6

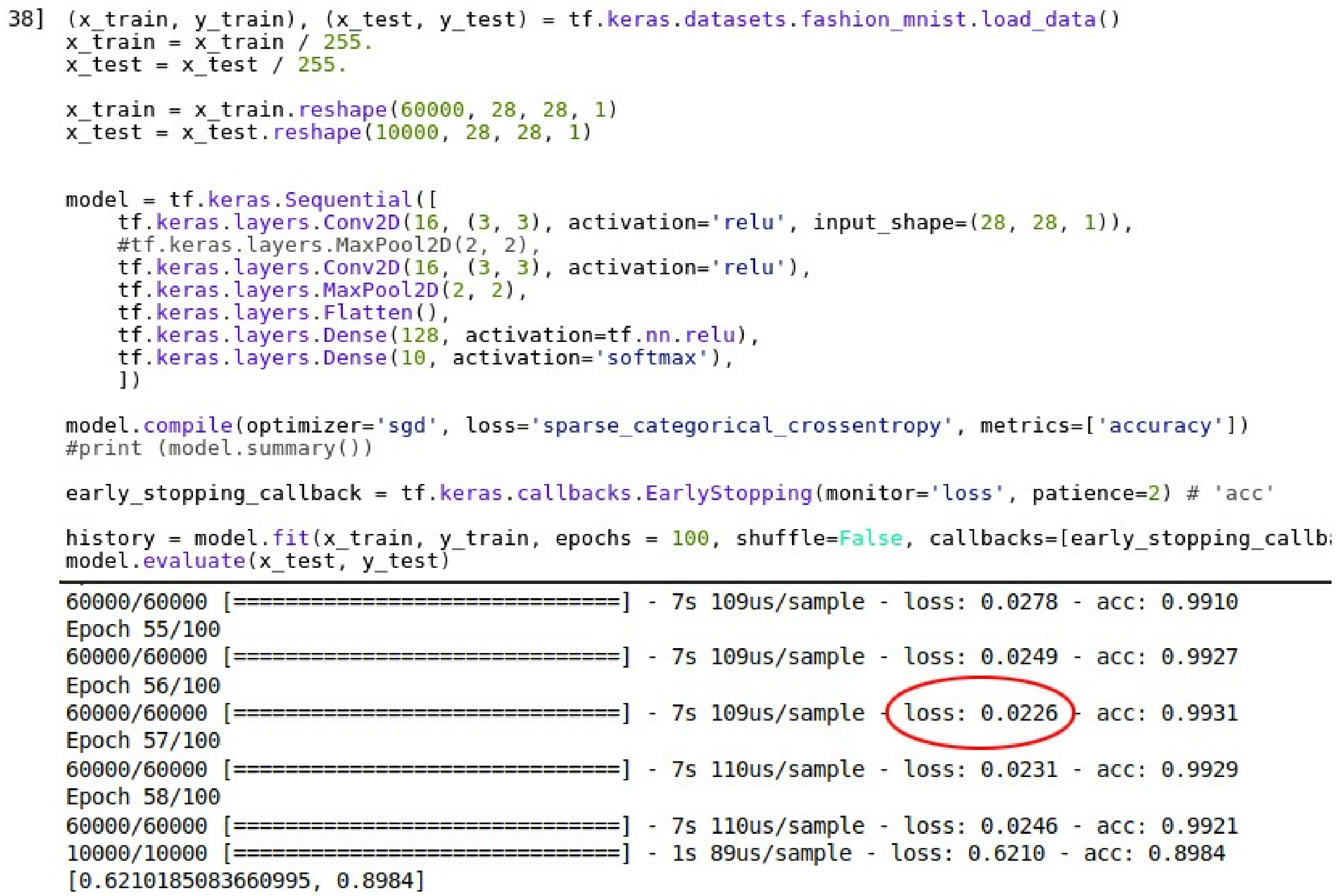
	Time(s)	Loss test	Accuracy test (%)	Loss train	Acc. train
Main model 1-0-1-1	184	0,300	90,67%	0,13%	95,83%



```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()
x_train = x_train / 255.
x_test = x_test / 255.

x_train = x_train.reshape(60000, 28, 28, 1)
x_test = x_test.reshape(10000, 28, 28, 1)


model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    #tf.keras.layers.MaxPool2D(2, 2),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(2, 2),
    #tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    #tf.keras.layers.MaxPool2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation=tf.nn.relu),
    #tf.keras.layers.Dropout (0.25),
    tf.keras.layers.Dense(10, activation='softmax'),
    ])

model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
print (model.summary())

early_stopping_callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=2)

history = model.fit(x_train, y_train, epochs = 1, shuffle=False, callbacks=[early_stopping_callback])
model.evaluate(x_test, y_test)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 32)        9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 12, 12, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 128)               589952
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 600,810
Trainable params: 600,810
Non-trainable params: 0
_________________________________________________________________
None
```

Implementation callback on the loss function



```
38] (x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()
    x_train = x_train / 255.
    x_test = x_test / 255.

    x_train = x_train.reshape(60000, 28, 28, 1)
    x_test = x_test.reshape(10000, 28, 28, 1)


    model = tf.keras.Sequential([
        tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(28, 28, 1)),
        #tf.keras.layers.MaxPool2D(2, 2),
        tf.keras.layers.Conv2D(16, (3, 3), activation='relu'),
        tf.keras.layers.MaxPool2D(2, 2),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation=tf.nn.relu),
        tf.keras.layers.Dense(10, activation='softmax'),
        ])

    model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    #print (model.summary())

    early_stopping_callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=2) # 'acc'

    history = model.fit(x_train, y_train, epochs = 100, shuffle=False, callbacks=[early_stopping_callb
    model.evaluate(x_test, y_test)
```

```
60000/60000 [==============================] - 7s 109us/sample - loss: 0.0278 - acc: 0.9910
Epoch 55/100
60000/60000 [==============================] - 7s 109us/sample - loss: 0.0249 - acc: 0.9927
Epoch 56/100
60000/60000 [==============================] - 7s 109us/sample - loss: 0.0226 - acc: 0.9931
Epoch 57/100
60000/60000 [==============================] - 7s 110us/sample - loss: 0.0231 - acc: 0.9929
Epoch 58/100
60000/60000 [==============================] - 7s 110us/sample - loss: 0.0246 - acc: 0.9921
10000/10000 [==============================] - 1s 89us/sample - loss: 0.6210 - acc: 0.8984
[0.6210185083660995, 0.8984]
```

After the implementation of callbacks and using the splitting of the training set into training and validation set, the optimal number of epochs was 16.

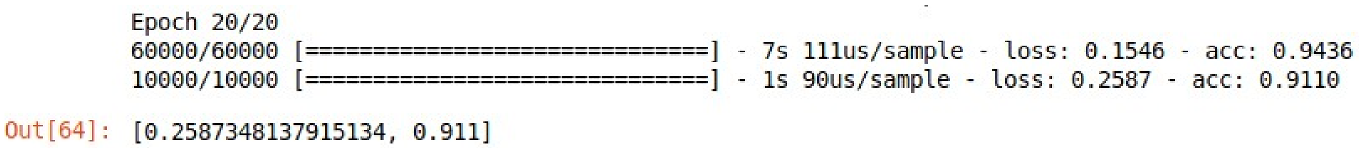
So, one last neural network training was conducted, and the result was obtained:

Other parameters of neural network:

feature maps = 32, shuffle= True

tab 7

	Epochs	Time(s)	Loss test	Acc. test	Loss train	Acc. train
Main model 1-1-1-1	**5**	33	0,418	84,79%	0,410	85,00%
Main model 1-0-1-1	**20**	184	0,259	91,10%	0,155	94,36%



```
         Epoch 20/20
         60000/60000 [==============================] - 7s 111us/sample - loss: 0.1546 - acc: 0.9436
         10000/10000 [==============================] - 1s 90us/sample - loss: 0.2587 - acc: 0.9110
Out[64]: [0.2587348137915134, 0.911]
```