Bogumiła Walkowiak bogumila.walkowiak@grenoble-inp.org

Joachim Mąkowski joachim-kajetan.makowski@grenoble-inp.org

Intelligent Systems: Reasoning and Recognition

Recognizing Digits using Neural Networks

1. Introduction

The MNIST (Modified National Institute of Standards and Technology) dataset is a large collection of handwritten digits composed of 60,000 training images and 10,000 test images. The black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced gray-scale levels. Our task was to design and evaluate neural network architectures that can recognize hand-drawn digits using the grayscale this data set.

2. Data preparation

First of all, we downloaded MNIST data. We decided to combine train and test set provided by MNIST dataset and then we splited data into training set 90% and a test set 10%. In the further part of the project, we'll also create a validation set so the final split of the data will look like this: training data 80%, validating data 10% and testing data 10%.

In [86]:
```
import numpy as np
import tensorflow.compat.v1.keras.backend as K
import tensorflow as tf
from tensorflow import keras

from tensorflow.keras import layers
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score,roc_curve,auc

from sklearn.metrics import confusion_matrix, plot_confusion_matrix
#physical_devices = tf.config.list_physical_devices('GPU')
#tf.config.experimental.set_memory_growth(physical_devices[0], True)
```

In [95]:
```
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
```

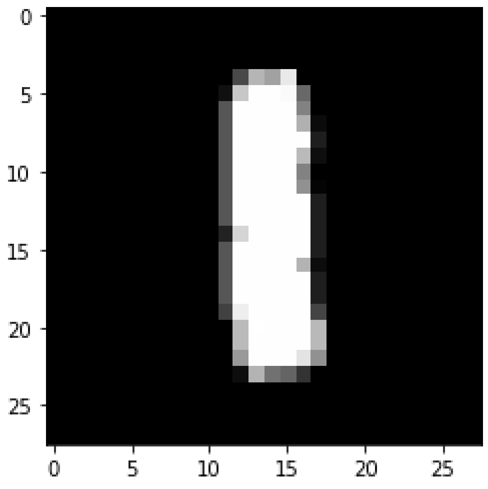
In [96]:
```
X = np.concatenate((x_train, x_test))
y = np.concatenate([y_train, y_test])
```

In [97]:
```
train_ratio = 0.9
test_ratio = 0.1
```

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = te
```

In [98]:
```
plt.imshow(x_train[0], cmap='gray')
```

Out[98]:
```
<matplotlib.image.AxesImage at 0x7f3cfc7c7b90>
```



In [99]:
```
x_train = x_train.astype("float32") / 255
x_test = x_test.astype("float32") / 255

# images have shape (28, 28, 1)
x_train = np.expand_dims(x_train, -1)
x_test = np.expand_dims(x_test, -1)
```

In [100…
```
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
```

3. Creating neural networks

We decided to create a function. Thanks to it we will be able to write less code. Function trains model provided by argument of function, prints model's loss, accuracy, precision, recall and AUC for each digit and plots a history of training.

In [60]:
```
def predict_model(model, callbacks = [],batch_size=128, epochs = 4,lr=0.
    adam = keras.optimizers.Adam(lr=lr)
    model.compile(loss="categorical_crossentropy", optimizer=adam, metri

    history = model.fit(x_train, y_train, batch_size=batch_size, epochs=
    score = model.evaluate(x_test, y_test, verbose=0)
    y_pred = model.predict(x_test)
    print("Test loss:", score[0])
    print("Test accuracy:", score[1])
    print("Test precision:", score[2])
    print("Test recall:", score[3])

    y_pred = np.argmax(y_pred,axis=1)
    y_test1 = np.argmax(y_test,axis=1)

    print("Test f1 score:", f1_score(y_test1,y_pred,average='micro'))
    for i in range(10):
        temp_pred = [1 if x==i else 0 for x in y_pred]
        temp_test = [1 if x==i else 0 for x in y_test1]
        fpr, tpr, thresholds =roc_curve(temp_test,temp_pred)

        print("Test AUC for digit:",i, auc(fpr, tpr))

    # summarize history for accuracy
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'val'], loc='upper left')
    plt.show()
    # summarize history for loss
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'val'], loc='upper left')
    plt.show()
```

We added an instance of EarlyStopping class, which provides us a mechanism of stopping algorithm before the whole training process is done. When 3 epochs are not achieving a better result (in our example higher validation accuracy) then our training is stopped and we restore the best model.

In [61]:
```
# simple early stopping
es = keras.callbacks.EarlyStopping(monitor='val_accuracy', mode='max', v
```

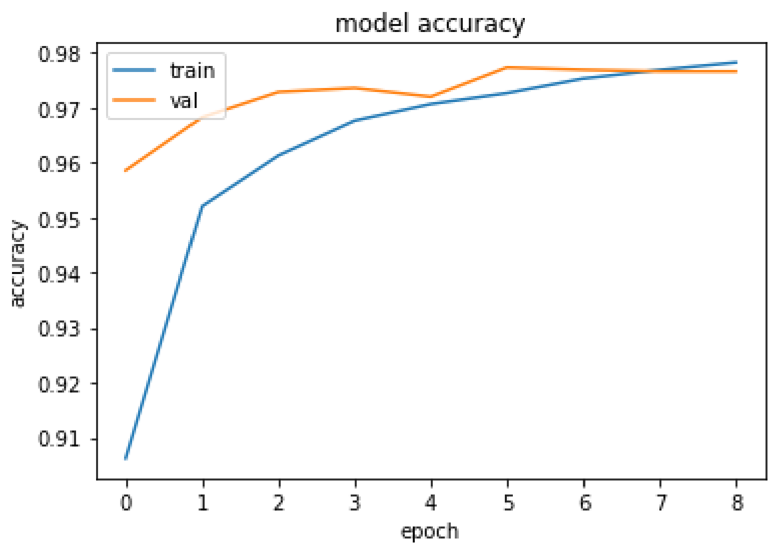
Basic Fully Connected Multi-layer Network

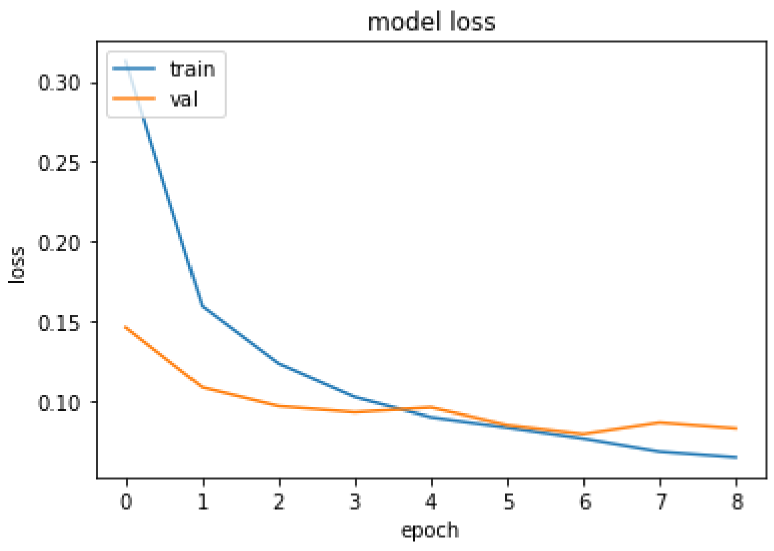
The first network we have created is basic fully connected mutli-layer network:

In [48]:
```
model_fc = keras.Sequential([
    layers.Dense(32, activation="relu",input_shape=(28,28,1)),
    layers.Dense(64, activation="relu"),

    layers.Flatten(),
    layers.Dense(128, activation="relu"),
    layers.Dropout(.25),
    layers.Dense(10, activation="softmax")
])
model_fc.summary()
predict_model(model_fc, [es], epochs=100)
```

```
Model: "sequential_20"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_59 (Dense)             (None, 28, 28, 32)        64        
_________________________________________________________________
dense_60 (Dense)             (None, 28, 28, 64)        2112      
_________________________________________________________________
flatten_20 (Flatten)         (None, 50176)             0         
_________________________________________________________________
dense_61 (Dense)             (None, 128)               6422656   
_________________________________________________________________
dropout_19 (Dropout)         (None, 128)               0         
_________________________________________________________________
```

```
dense_62 (Dense)            (None, 10)               1290
=================================================================
Total params: 6,426,122
Trainable params: 6,426,122
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
439/439 [==============================] - 6s 13ms/step - loss: 0.5066 -
accuracy: 0.8487 - precision: 0.9149 - recall: 0.7786 - val_loss: 0.1463
- val_accuracy: 0.9586 - val_precision: 0.9667 - val_recall: 0.9504
Epoch 2/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1643 -
accuracy: 0.9510 - precision: 0.9601 - recall: 0.9434 - val_loss: 0.1087
- val_accuracy: 0.9683 - val_precision: 0.9707 - val_recall: 0.9661
Epoch 3/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1254 -
accuracy: 0.9611 - precision: 0.9680 - recall: 0.9554 - val_loss: 0.0970
- val_accuracy: 0.9729 - val_precision: 0.9770 - val_recall: 0.9698
Epoch 4/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1044 -
accuracy: 0.9666 - precision: 0.9716 - recall: 0.9629 - val_loss: 0.0932
- val_accuracy: 0.9736 - val_precision: 0.9756 - val_recall: 0.9711
Epoch 5/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0893 -
accuracy: 0.9704 - precision: 0.9740 - recall: 0.9670 - val_loss: 0.0962
- val_accuracy: 0.9720 - val_precision: 0.9748 - val_recall: 0.9710
Epoch 6/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0824 -
accuracy: 0.9726 - precision: 0.9757 - recall: 0.9695 - val_loss: 0.0853
- val_accuracy: 0.9773 - val_precision: 0.9801 - val_recall: 0.9746
Epoch 7/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0750 -
accuracy: 0.9763 - precision: 0.9789 - recall: 0.9733 - val_loss: 0.0796
- val_accuracy: 0.9769 - val_precision: 0.9796 - val_recall: 0.9755
Epoch 8/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0664 -
accuracy: 0.9776 - precision: 0.9805 - recall: 0.9752 - val_loss: 0.0869
- val_accuracy: 0.9766 - val_precision: 0.9787 - val_recall: 0.9755
Epoch 9/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0621 -
accuracy: 0.9788 - precision: 0.9806 - recall: 0.9770 - val_loss: 0.0832
- val_accuracy: 0.9766 - val_precision: 0.9783 - val_recall: 0.9758
Restoring model weights from the end of the best epoch.
Epoch 00009: early stopping
Test loss: 0.09832347929477692
Test accuracy: 0.973714292049408
Test precision: 0.9774295687675476
Test recall: 0.9712857007980347
Test f1 score: 0.9737142857142858
Test AUC for digit: 0 0.9907399351644153
Test AUC for digit: 1 0.9923331284991935
Test AUC for digit: 2 0.9870421148228989
Test AUC for digit: 3 0.9851853814963031
Test AUC for digit: 4 0.9845406523282492
Test AUC for digit: 5 0.9781305833322657
Test AUC for digit: 6 0.9906349206349208
Test AUC for digit: 7 0.9875151552516612
Test AUC for digit: 8 0.974920634920635
Test AUC for digit: 9 0.9815057646170433
```





This is basic model achieves about 97,5% accuracy on test set. It is made of 2 hidden layers with reasonable number of units. Training this model is quite fast (on my laptop it was 5s per epoch, using GPU). As we see in plots our model started to overfits, because validation accuracy and loss was staying on the same level, while train accuracy was growing and loss was decreasing.

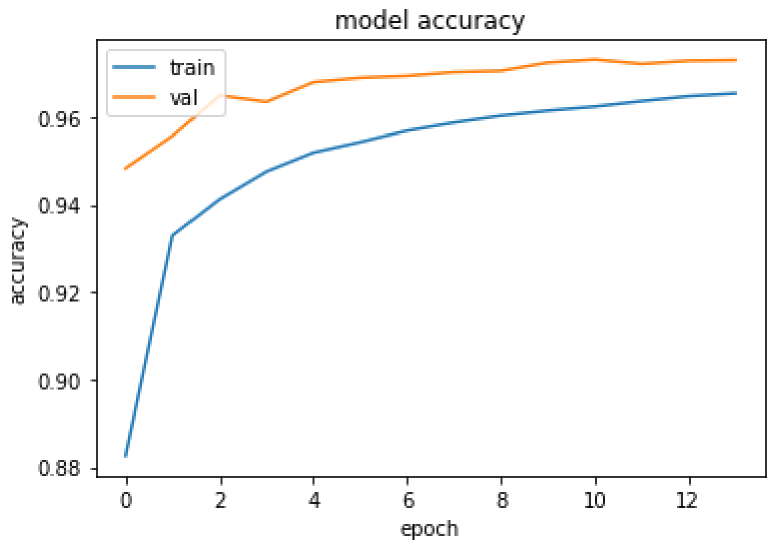
Next, we wanted to demonstrate the effect of changing various parameters of the network.

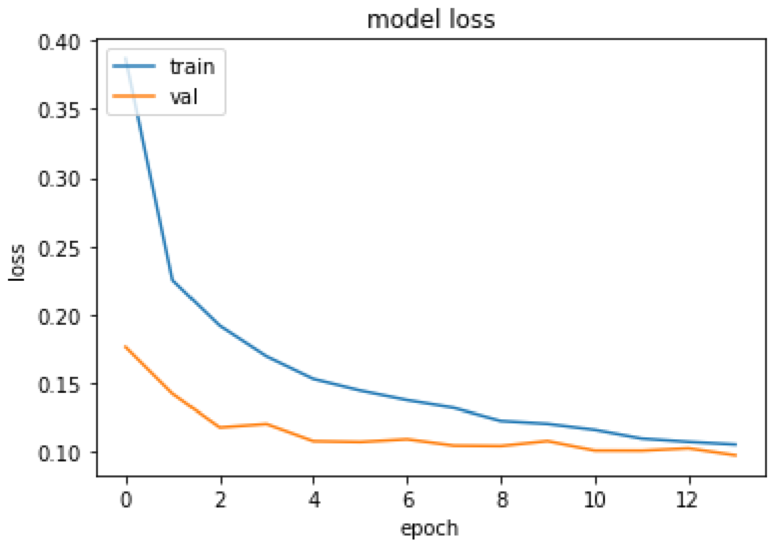
Different number of layers

In [13]:
```
model_fc_small = keras.Sequential([
    layers.Dense(32, activation="relu",input_shape=(28,28,1)),

    layers.Flatten(),
    layers.Dense(64, activation="relu"),
    layers.Dropout(.25),
    layers.Dense(10, activation="softmax")
])
model_fc_small.summary()
predict_model(model_fc_small, [es], epochs=100)
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_4 (Dense)              (None, 28, 28, 32)        64
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_5 (Dense)              (None, 64)                1605696
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_6 (Dense)              (None, 10)                650
=================================================================
Total params: 1,606,410
Trainable params: 1,606,410
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
439/439 [==============================] - 11s 22ms/step - loss: 0.5918 -
accuracy: 0.8171 - precision: 0.9070 - recall: 0.7388 - val_loss: 0.1764
- val_accuracy: 0.9483 - val_precision: 0.9603 - val_recall: 0.9381
Epoch 2/100
439/439 [==============================] - 9s 21ms/step - loss: 0.2418 -
accuracy: 0.9282 - precision: 0.9433 - recall: 0.9133 - val_loss: 0.1424
- val_accuracy: 0.9557 - val_precision: 0.9640 - val_recall: 0.9494
Epoch 3/100
439/439 [==============================] - 10s 22ms/step - loss: 0.1958 -
accuracy: 0.9404 - precision: 0.9512 - recall: 0.9299 - val_loss: 0.1180
- val_accuracy: 0.9649 - val_precision: 0.9711 - val_recall: 0.9587
Epoch 4/100
439/439 [==============================] - 9s 21ms/step - loss: 0.1628 -
accuracy: 0.9478 - precision: 0.9577 - recall: 0.9405 - val_loss: 0.1205
- val_accuracy: 0.9635 - val_precision: 0.9701 - val_recall: 0.9589
Epoch 5/100
439/439 [==============================] - 9s 22ms/step - loss: 0.1513 -
accuracy: 0.9516 - precision: 0.9591 - recall: 0.9447 - val_loss: 0.1079
- val_accuracy: 0.9680 - val_precision: 0.9741 - val_recall: 0.9644
Epoch 6/100
439/439 [==============================] - 9s 22ms/step - loss: 0.1431 -
accuracy: 0.9555 - precision: 0.9634 - recall: 0.9489 - val_loss: 0.1074
- val_accuracy: 0.9690 - val_precision: 0.9747 - val_recall: 0.9674
Epoch 7/100
439/439 [==============================] - 10s 22ms/step - loss: 0.1351 -
accuracy: 0.9565 - precision: 0.9637 - recall: 0.9509 - val_loss: 0.1093
- val_accuracy: 0.9694 - val_precision: 0.9741 - val_recall: 0.9646
Epoch 8/100
439/439 [==============================] - 10s 22ms/step - loss: 0.1319 -
accuracy: 0.9587 - precision: 0.9651 - recall: 0.9528 - val_loss: 0.1047
- val_accuracy: 0.9703 - val_precision: 0.9741 - val_recall: 0.9674
Epoch 9/100
439/439 [==============================] - 10s 22ms/step - loss: 0.1188 -
accuracy: 0.9614 - precision: 0.9674 - recall: 0.9562 - val_loss: 0.1044
- val_accuracy: 0.9706 - val_precision: 0.9747 - val_recall: 0.9670
Epoch 10/100
439/439 [==============================] - 10s 22ms/step - loss: 0.1194 -
accuracy: 0.9610 - precision: 0.9666 - recall: 0.9566 - val_loss: 0.1080
- val_accuracy: 0.9724 - val_precision: 0.9749 - val_recall: 0.9697
Epoch 11/100
439/439 [==============================] - 10s 22ms/step - loss: 0.1116 -
accuracy: 0.9636 - precision: 0.9691 - recall: 0.9591 - val_loss: 0.1011
- val_accuracy: 0.9732 - val_precision: 0.9760 - val_recall: 0.9693
Epoch 12/100
439/439 [==============================] - 10s 22ms/step - loss: 0.1030 -
accuracy: 0.9653 - precision: 0.9705 - recall: 0.9611 - val_loss: 0.1011
- val_accuracy: 0.9722 - val_precision: 0.9743 - val_recall: 0.9698
Epoch 13/100
439/439 [==============================] - 10s 22ms/step - loss: 0.1039 -
accuracy: 0.9663 - precision: 0.9707 - recall: 0.9623 - val_loss: 0.1028
- val_accuracy: 0.9729 - val_precision: 0.9764 - val_recall: 0.9713
Epoch 14/100
439/439 [==============================] - 10s 22ms/step - loss: 0.1029 -
```

```
accuracy: 0.9668 - precision: 0.9710 - recall: 0.9625 - val_loss: 0.0977
- val_accuracy: 0.9730 - val_precision: 0.9772 - val_recall: 0.9717
Epoch 00014: early stopping
Test loss: 0.1229480430483818
Test accuracy: 0.9679999947547913
Test precision: 0.9723661541938782
Test recall: 0.9651428461074829
Test f1 score: 0.968
Test AUC for digit: 0 0.9900292592326402
Test AUC for digit: 1 0.9906050941195285
Test AUC for digit: 2 0.9857744868717027
Test AUC for digit: 3 0.9805160602125461
Test AUC for digit: 4 0.981489857731353
Test AUC for digit: 5 0.9728804648288869
Test AUC for digit: 6 0.9881746031746033
Test AUC for digit: 7 0.9815794883823452
Test AUC for digit: 8 0.9792063492063492
Test AUC for digit: 9 0.9701437036094706
```



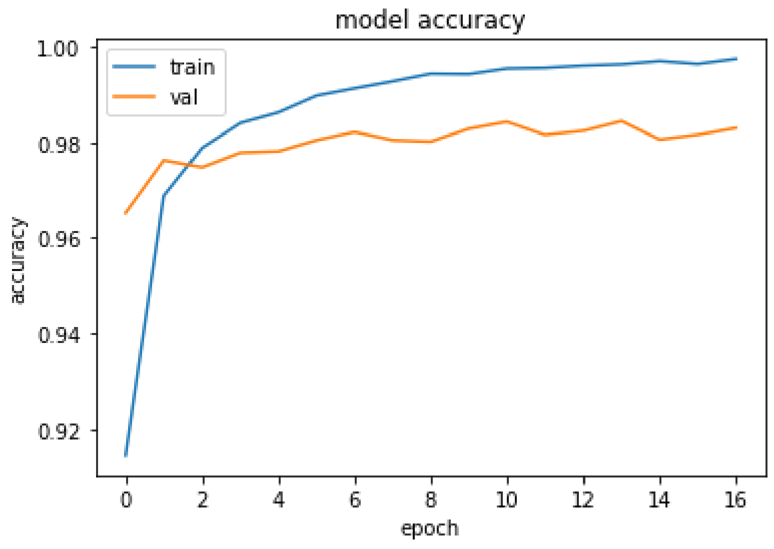


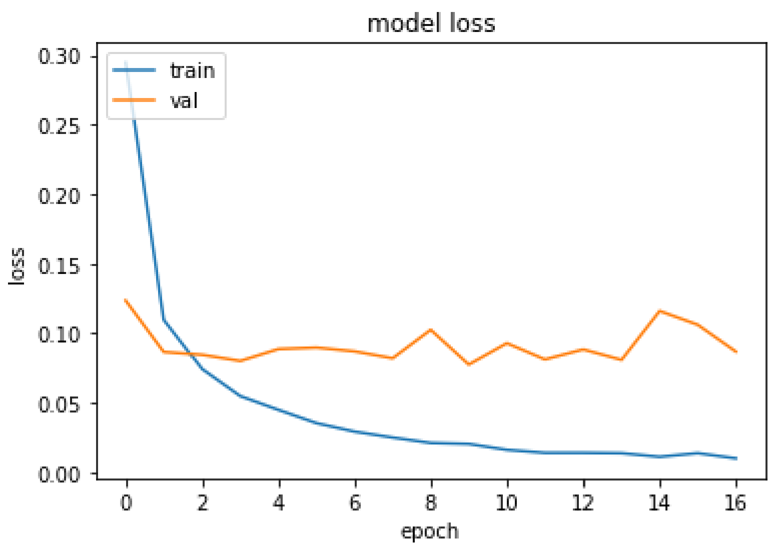
In [14]:
```python
model_fc_large = keras.Sequential([
    layers.Dense(32, activation="relu",input_shape=(28,28,1)),
    layers.Dense(64, activation="relu"),
    layers.Flatten(),
    layers.Dense(4096, activation="relu"),
    layers.Dense(1024, activation="relu"),
    layers.Dense(64, activation="relu"),
    layers.Dropout(.25),
    layers.Dense(10, activation="softmax")
])
```

```
model_fc_large.summary()
predict_model(model_fc_large, [es], epochs=100)
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 28, 28, 32)        64
_________________________________________________________________
dense_8 (Dense)              (None, 28, 28, 64)        2112
_________________________________________________________________
flatten_2 (Flatten)          (None, 50176)             0
_________________________________________________________________
dense_9 (Dense)              (None, 4096)              205524992
_________________________________________________________________
dense_10 (Dense)             (None, 1024)              4195328
_________________________________________________________________
dense_11 (Dense)             (None, 64)                65600
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_12 (Dense)             (None, 10)                650
=================================================================
Total params: 209,788,746
Trainable params: 209,788,746
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
439/439 [==============================] - 55s 120ms/step - loss: 0.5493
- accuracy: 0.8354 - precision: 0.9044 - recall: 0.7825 - val_loss: 0.123
6 - val_accuracy: 0.9652 - val_precision: 0.9722 - val_recall: 0.9605
Epoch 2/100
439/439 [==============================] - 33s 75ms/step - loss: 0.1123 -
accuracy: 0.9679 - precision: 0.9741 - recall: 0.9620 - val_loss: 0.0866
- val_accuracy: 0.9762 - val_precision: 0.9798 - val_recall: 0.9717
Epoch 3/100
439/439 [==============================] - 33s 75ms/step - loss: 0.0735 -
accuracy: 0.9787 - precision: 0.9823 - recall: 0.9749 - val_loss: 0.0845
- val_accuracy: 0.9747 - val_precision: 0.9812 - val_recall: 0.9709
Epoch 4/100
439/439 [==============================] - 33s 75ms/step - loss: 0.0509 -
accuracy: 0.9847 - precision: 0.9870 - recall: 0.9826 - val_loss: 0.0801
- val_accuracy: 0.9778 - val_precision: 0.9823 - val_recall: 0.9752
Epoch 5/100
439/439 [==============================] - 33s 75ms/step - loss: 0.0464 -
accuracy: 0.9860 - precision: 0.9879 - recall: 0.9842 - val_loss: 0.0887
- val_accuracy: 0.9781 - val_precision: 0.9806 - val_recall: 0.9762
Epoch 6/100
439/439 [==============================] - 33s 75ms/step - loss: 0.0356 -
accuracy: 0.9898 - precision: 0.9909 - recall: 0.9882 - val_loss: 0.0897
- val_accuracy: 0.9804 - val_precision: 0.9828 - val_recall: 0.9785
Epoch 7/100
439/439 [==============================] - 33s 75ms/step - loss: 0.0276 -
accuracy: 0.9921 - precision: 0.9929 - recall: 0.9913 - val_loss: 0.0870
- val_accuracy: 0.9821 - val_precision: 0.9832 - val_recall: 0.9811
Epoch 8/100
439/439 [==============================] - 33s 75ms/step - loss: 0.0234 -
accuracy: 0.9928 - precision: 0.9935 - recall: 0.9920 - val_loss: 0.0820
- val_accuracy: 0.9804 - val_precision: 0.9823 - val_recall: 0.9776
Epoch 9/100
439/439 [==============================] - 33s 75ms/step - loss: 0.0215 -
accuracy: 0.9939 - precision: 0.9947 - recall: 0.9931 - val_loss: 0.1025
- val_accuracy: 0.9801 - val_precision: 0.9812 - val_recall: 0.9786
Epoch 10/100
439/439 [==============================] - 32s 73ms/step - loss: 0.0220 -
accuracy: 0.9939 - precision: 0.9943 - recall: 0.9933 - val_loss: 0.0775
- val_accuracy: 0.9828 - val_precision: 0.9849 - val_recall: 0.9821
Epoch 11/100
```

```
439/439 [==============================] - 34s 77ms/step - loss: 0.0177 -
accuracy: 0.9949 - precision: 0.9953 - recall: 0.9944 - val_loss: 0.0929
- val_accuracy: 0.9843 - val_precision: 0.9850 - val_recall: 0.9831
Epoch 12/100
439/439 [==============================] - 34s 79ms/step - loss: 0.0138 -
accuracy: 0.9958 - precision: 0.9962 - recall: 0.9955 - val_loss: 0.0812
- val_accuracy: 0.9815 - val_precision: 0.9828 - val_recall: 0.9808
Epoch 13/100
439/439 [==============================] - 34s 77ms/step - loss: 0.0136 -
accuracy: 0.9963 - precision: 0.9965 - recall: 0.9960 - val_loss: 0.0883
- val_accuracy: 0.9824 - val_precision: 0.9834 - val_recall: 0.9818
Epoch 14/100
439/439 [==============================] - 34s 77ms/step - loss: 0.0117 -
accuracy: 0.9965 - precision: 0.9967 - recall: 0.9963 - val_loss: 0.0809
- val_accuracy: 0.9844 - val_precision: 0.9858 - val_recall: 0.9844
Epoch 15/100
439/439 [==============================] - 34s 78ms/step - loss: 0.0107 -
accuracy: 0.9971 - precision: 0.9972 - recall: 0.9968 - val_loss: 0.1161
- val_accuracy: 0.9805 - val_precision: 0.9811 - val_recall: 0.9798
Epoch 16/100
439/439 [==============================] - 33s 74ms/step - loss: 0.0152 -
accuracy: 0.9959 - precision: 0.9960 - recall: 0.9955 - val_loss: 0.1062
- val_accuracy: 0.9815 - val_precision: 0.9827 - val_recall: 0.9814
Epoch 17/100
439/439 [==============================] - 34s 78ms/step - loss: 0.0097 -
accuracy: 0.9971 - precision: 0.9973 - recall: 0.9970 - val_loss: 0.0868
- val_accuracy: 0.9830 - val_precision: 0.9841 - val_recall: 0.9824
Epoch 00017: early stopping
Test loss: 0.09394106268882751
Test accuracy: 0.9814285635948181
Test precision: 0.9819768071174622
Test recall: 0.9807142615318298
Test f1 score: 0.9814285714285714
Test AUC for digit: 0 0.9988155401137082
Test AUC for digit: 1 0.9947057004231706
Test AUC for digit: 2 0.9863037767355775
Test AUC for digit: 3 0.9898547027800599
Test AUC for digit: 4 0.9875124205303294
Test AUC for digit: 5 0.9887902209336588
Test AUC for digit: 6 0.9899999999999999
Test AUC for digit: 7 0.9909183219141172
Test AUC for digit: 8 0.9815079365079366
Test AUC for digit: 9 0.987926026320382
```





Firstly, we tried different numbers of hidden layers. With 1 hidden layer the model the model was achieving around 96,5% on test set. The model is underfitted because this number of layers is not enough to explain the complexity of our data.

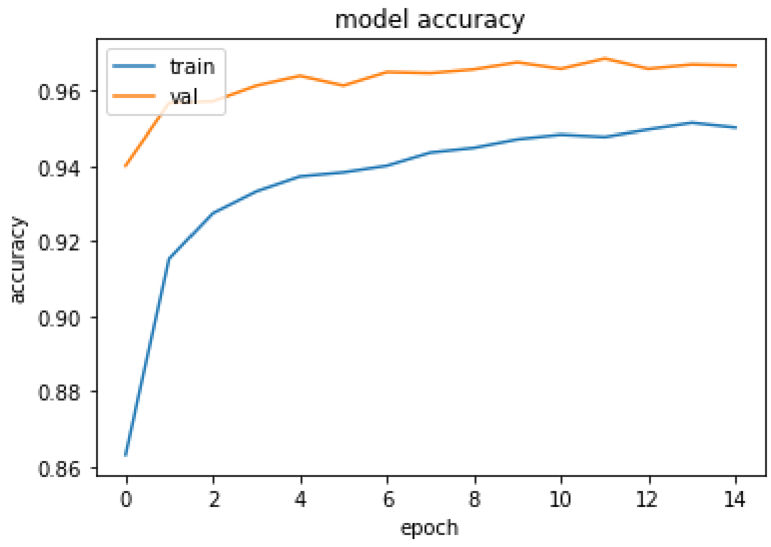
Model with 4 hidden layers achieved 98,1% of accuracy but the training time was pretty long (34s per epoch). That is because this model had to find weights for over 200,000,000 parameters (compering to 1,600,000 of params for model with 1 hidden layer). We can assume, that after second epoch our model is overfitted because the difference between validation and train loss and accuracy are high.

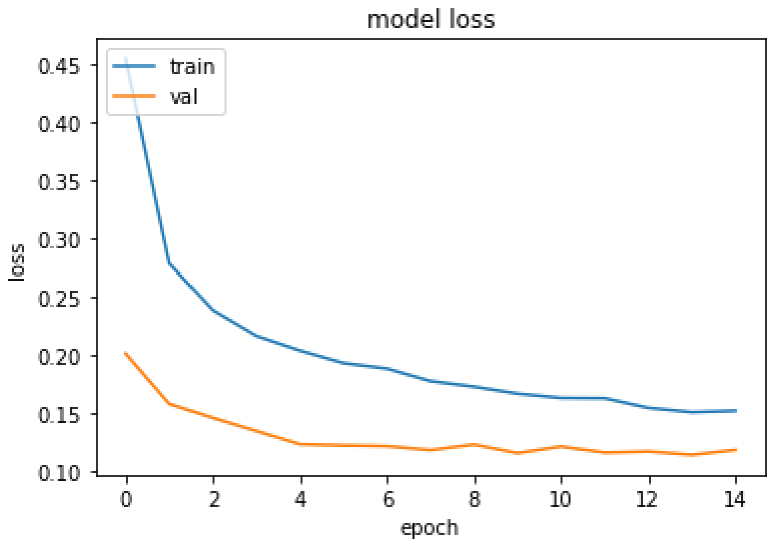
Different number of units per layer

In [15]:
```
model_fc = keras.Sequential([
    layers.Dense(10, activation="relu",input_shape=(28,28,1)),
    layers.Dense(20, activation="relu"),

    layers.Flatten(),
    layers.Dense(40, activation="relu"),
    layers.Dropout(.25),
    layers.Dense(10, activation="softmax")
])
model_fc.summary()
predict_model(model_fc, [es], epochs=100)
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_13 (Dense)             (None, 28, 28, 10)        20
_________________________________________________________________
dense_14 (Dense)             (None, 28, 28, 20)        220
_________________________________________________________________
flatten_3 (Flatten)          (None, 15680)             0
_________________________________________________________________
dense_15 (Dense)             (None, 40)                627240
_________________________________________________________________
dropout_3 (Dropout)          (None, 40)                0
_________________________________________________________________
dense_16 (Dense)             (None, 10)                410
=================================================================
Total params: 627,890
Trainable params: 627,890
Non-trainable params: 0
```

```
_________________________________________________________________
Epoch 1/100
439/439 [==============================] - 4s 8ms/step - loss: 0.6942 - a
ccuracy: 0.7885 - precision: 0.8959 - recall: 0.6762 - val_loss: 0.2011 -
val_accuracy: 0.9401 - val_precision: 0.9534 - val_recall: 0.9297
Epoch 2/100
439/439 [==============================] - 3s 7ms/step - loss: 0.2938 - a
ccuracy: 0.9101 - precision: 0.9330 - recall: 0.8907 - val_loss: 0.1577 -
val_accuracy: 0.9567 - val_precision: 0.9660 - val_recall: 0.9476
Epoch 3/100
439/439 [==============================] - 3s 7ms/step - loss: 0.2441 - a
ccuracy: 0.9252 - precision: 0.9418 - recall: 0.9100 - val_loss: 0.1455 -
val_accuracy: 0.9571 - val_precision: 0.9666 - val_recall: 0.9519
Epoch 4/100
439/439 [==============================] - 3s 7ms/step - loss: 0.2146 - a
ccuracy: 0.9344 - precision: 0.9488 - recall: 0.9216 - val_loss: 0.1342 -
val_accuracy: 0.9613 - val_precision: 0.9710 - val_recall: 0.9534
Epoch 5/100
439/439 [==============================] - 3s 7ms/step - loss: 0.2027 - a
ccuracy: 0.9370 - precision: 0.9503 - recall: 0.9245 - val_loss: 0.1228 -
val_accuracy: 0.9639 - val_precision: 0.9711 - val_recall: 0.9589
Epoch 6/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1886 - a
ccuracy: 0.9395 - precision: 0.9515 - recall: 0.9284 - val_loss: 0.1218 -
val_accuracy: 0.9613 - val_precision: 0.9689 - val_recall: 0.9582
Epoch 7/100
439/439 [==============================] - 3s 8ms/step - loss: 0.1840 - a
ccuracy: 0.9408 - precision: 0.9528 - recall: 0.9316 - val_loss: 0.1211 -
val_accuracy: 0.9649 - val_precision: 0.9713 - val_recall: 0.9612
Epoch 8/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1801 - a
ccuracy: 0.9424 - precision: 0.9536 - recall: 0.9319 - val_loss: 0.1177 -
val_accuracy: 0.9646 - val_precision: 0.9717 - val_recall: 0.9613
Epoch 9/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1690 - a
ccuracy: 0.9457 - precision: 0.9563 - recall: 0.9362 - val_loss: 0.1225 -
val_accuracy: 0.9657 - val_precision: 0.9714 - val_recall: 0.9615
Epoch 10/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1685 - a
ccuracy: 0.9465 - precision: 0.9569 - recall: 0.9375 - val_loss: 0.1151 -
val_accuracy: 0.9675 - val_precision: 0.9729 - val_recall: 0.9644
Epoch 11/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1615 - a
ccuracy: 0.9484 - precision: 0.9589 - recall: 0.9397 - val_loss: 0.1208 -
val_accuracy: 0.9658 - val_precision: 0.9719 - val_recall: 0.9596
Epoch 12/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1573 - a
ccuracy: 0.9494 - precision: 0.9584 - recall: 0.9422 - val_loss: 0.1155 -
val_accuracy: 0.9685 - val_precision: 0.9716 - val_recall: 0.9641
Epoch 13/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1518 - a
ccuracy: 0.9498 - precision: 0.9592 - recall: 0.9414 - val_loss: 0.1165 -
val_accuracy: 0.9658 - val_precision: 0.9719 - val_recall: 0.9616
Epoch 14/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1459 - a
ccuracy: 0.9534 - precision: 0.9620 - recall: 0.9461 - val_loss: 0.1136 -
val_accuracy: 0.9670 - val_precision: 0.9723 - val_recall: 0.9641
Epoch 15/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1485 - a
ccuracy: 0.9509 - precision: 0.9603 - recall: 0.9437 - val_loss: 0.1178 -
val_accuracy: 0.9667 - val_precision: 0.9704 - val_recall: 0.9641
Epoch 00015: early stopping
Test loss: 0.12980997562408447
Test accuracy: 0.9620000123977661
Test precision: 0.9678164124488831
Test recall: 0.9580000042915344
Test f1 score: 0.962
Test AUC for digit: 0 0.9925116601919346
Test AUC for digit: 1 0.9898799893821776
Test AUC for digit: 2 0.9862624960357879
Test AUC for digit: 3 0.9697247767206424
Test AUC for digit: 4 0.9822645042790503
Test AUC for digit: 5 0.9736597292505758
Test AUC for digit: 6 0.9884920634920635
Test AUC for digit: 7 0.9744171471575807
Test AUC for digit: 8 0.9637301587301589
Test AUC for digit: 9 0.9673563983253335
```





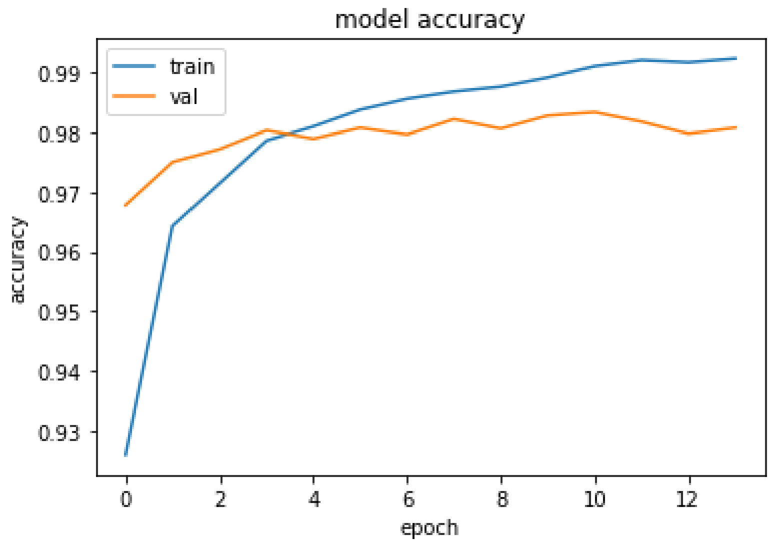
In this situation we trained a model with small number of units in each layer. The model didn't achieve it's best. We can see, that train accuracy is much lower than validation accuracy. It is caused by insufficient number of units, so that our model decided to choose higher accuracy in validation data at the expense of accuracy on whole data.

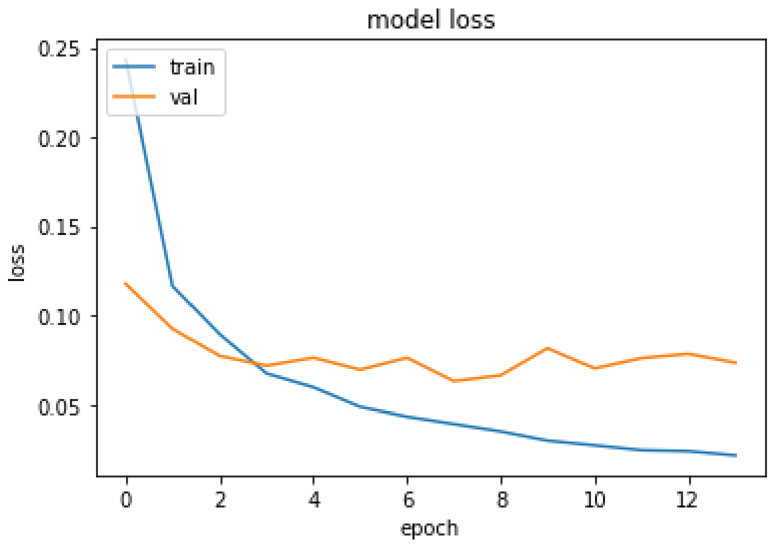
In [16]:

```
model_fc = keras.Sequential([
    layers.Dense(100, activation="relu",input_shape=(28,28,1)),
    layers.Dense(200, activation="relu"),

    layers.Flatten(),
    layers.Dense(400, activation="relu"),
    layers.Dropout(.25),
    layers.Dense(10, activation="softmax")
])
model_fc.summary()
predict_model(model_fc, [es], epochs=100)
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_17 (Dense)             (None, 28, 28, 100)       200       
_________________________________________________________________
dense_18 (Dense)             (None, 28, 28, 200)       20200     
_________________________________________________________________
flatten_4 (Flatten)          (None, 156800)            0         
_________________________________________________________________
dense_19 (Dense)             (None, 400)               62720400  
_________________________________________________________________
dropout_4 (Dropout)          (None, 400)               0         
_________________________________________________________________
dense_20 (Dense)             (None, 10)                4010      
=================================================================
Total params: 62,744,810
Trainable params: 62,744,810
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
439/439 [==============================] - 20s 45ms/step - loss: 0.3810 -
accuracy: 0.8801 - precision: 0.9206 - recall: 0.8528 - val_loss: 0.1177
- val_accuracy: 0.9678 - val_precision: 0.9722 - val_recall: 0.9632
Epoch 2/100
439/439 [==============================] - 19s 44ms/step - loss: 0.1207 -
accuracy: 0.9633 - precision: 0.9690 - recall: 0.9591 - val_loss: 0.0924
- val_accuracy: 0.9750 - val_precision: 0.9782 - val_recall: 0.9726
Epoch 3/100
439/439 [==============================] - 19s 44ms/step - loss: 0.0939 -
accuracy: 0.9698 - precision: 0.9735 - recall: 0.9668 - val_loss: 0.0772
- val_accuracy: 0.9772 - val_precision: 0.9801 - val_recall: 0.9756
Epoch 4/100
439/439 [==============================] - 19s 44ms/step - loss: 0.0664 -
accuracy: 0.9785 - precision: 0.9809 - recall: 0.9767 - val_loss: 0.0717
- val_accuracy: 0.9804 - val_precision: 0.9828 - val_recall: 0.9789
Epoch 5/100
439/439 [==============================] - 19s 44ms/step - loss: 0.0588 -
accuracy: 0.9814 - precision: 0.9831 - recall: 0.9800 - val_loss: 0.0762
- val_accuracy: 0.9789 - val_precision: 0.9824 - val_recall: 0.9771
Epoch 6/100
439/439 [==============================] - 19s 44ms/step - loss: 0.0463 -
accuracy: 0.9847 - precision: 0.9860 - recall: 0.9832 - val_loss: 0.0695
- val_accuracy: 0.9808 - val_precision: 0.9831 - val_recall: 0.9798
Epoch 7/100
439/439 [==============================] - 19s 44ms/step - loss: 0.0436 -
accuracy: 0.9851 - precision: 0.9862 - recall: 0.9839 - val_loss: 0.0762
- val_accuracy: 0.9797 - val_precision: 0.9813 - val_recall: 0.9784
Epoch 8/100
439/439 [==============================] - 19s 44ms/step - loss: 0.0367 -
accuracy: 0.9878 - precision: 0.9884 - recall: 0.9864 - val_loss: 0.0631
- val_accuracy: 0.9823 - val_precision: 0.9836 - val_recall: 0.9808
Epoch 9/100
439/439 [==============================] - 19s 44ms/step - loss: 0.0343 -
accuracy: 0.9876 - precision: 0.9886 - recall: 0.9868 - val_loss: 0.0663
- val_accuracy: 0.9807 - val_precision: 0.9830 - val_recall: 0.9782
Epoch 10/100
439/439 [==============================] - 19s 44ms/step - loss: 0.0300 -
accuracy: 0.9894 - precision: 0.9902 - recall: 0.9889 - val_loss: 0.0815
- val_accuracy: 0.9828 - val_precision: 0.9831 - val_recall: 0.9820
Epoch 11/100
439/439 [==============================] - 19s 44ms/step - loss: 0.0253 -
accuracy: 0.9917 - precision: 0.9922 - recall: 0.9912 - val_loss: 0.0702
- val_accuracy: 0.9834 - val_precision: 0.9847 - val_recall: 0.9830
Epoch 12/100
439/439 [==============================] - 19s 44ms/step - loss: 0.0224 -
accuracy: 0.9931 - precision: 0.9933 - recall: 0.9928 - val_loss: 0.0760
- val_accuracy: 0.9818 - val_precision: 0.9831 - val_recall: 0.9811
```

```
Epoch 13/100
439/439 [==============================] - 19s 44ms/step - loss: 0.0239 -
accuracy: 0.9915 - precision: 0.9921 - recall: 0.9910 - val_loss: 0.0783
- val_accuracy: 0.9798 - val_precision: 0.9818 - val_recall: 0.9789
Epoch 14/100
439/439 [==============================] - 19s 44ms/step - loss: 0.0202 -
accuracy: 0.9927 - precision: 0.9931 - recall: 0.9925 - val_loss: 0.0734
- val_accuracy: 0.9808 - val_precision: 0.9828 - val_recall: 0.9795
Epoch 00014: early stopping
Test loss: 0.08533725142478943
Test accuracy: 0.9795714020729065
Test precision: 0.9812401533126831
Test recall: 0.978857159614563
Test f1 score: 0.9795714285714285
Test AUC for digit: 0 0.993617156085807
Test AUC for digit: 1 0.9955623103018043
Test AUC for digit: 2 0.9869502393935624
Test AUC for digit: 3 0.9921123419818196
Test AUC for digit: 4 0.9906739460192185
Test AUC for digit: 5 0.9834615107924303
Test AUC for digit: 6 0.993095238095238
Test AUC for digit: 7 0.990522564434301
Test AUC for digit: 8 0.976031746031746
Test AUC for digit: 9 0.9826922857371192
```





In this model we see that it's overfitting after third epoch. It is caused by too high number of units.

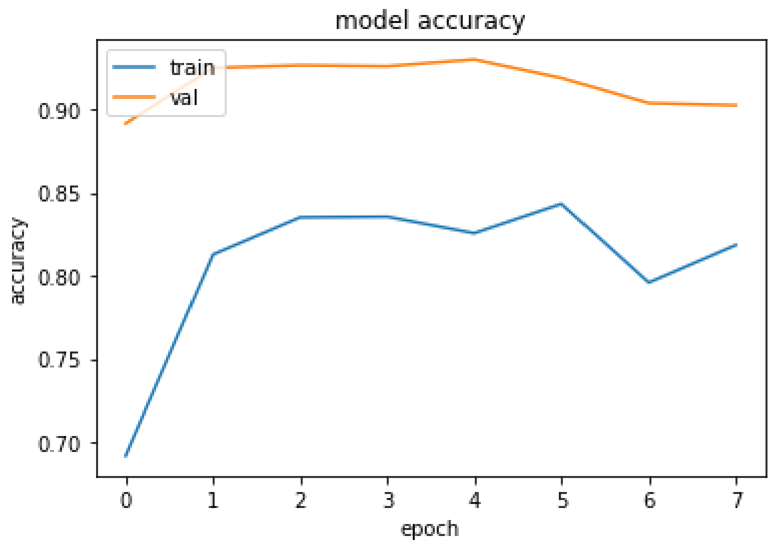
Different learning rate

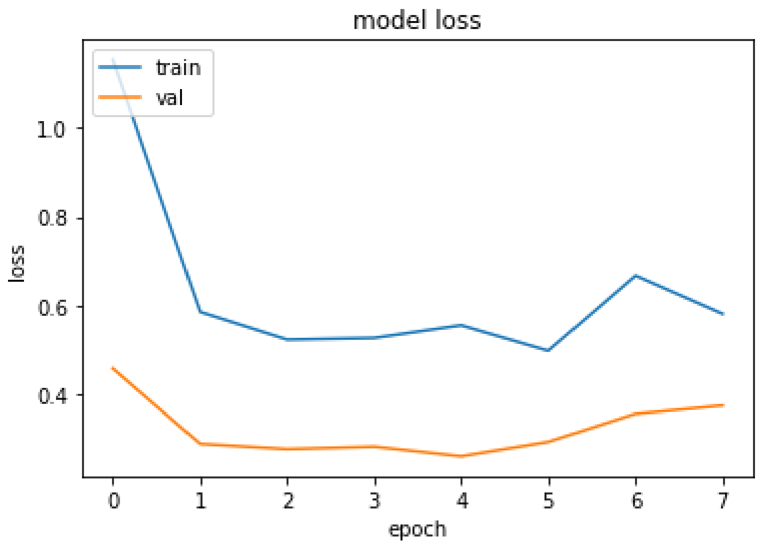
In [49]:

```
model_fc_01 = keras.Sequential([
    layers.Dense(32, activation="relu",input_shape=(28,28,1)),
    layers.Dense(64, activation="relu"),

    layers.Flatten(),
    layers.Dense(128, activation="relu"),
    layers.Dropout(.25),
    layers.Dense(10, activation="softmax")
])
model_fc_01.summary()
predict_model(model_fc_01,[es], epochs=100, lr=0.05)
```

```
Model: "sequential_21"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_63 (Dense)             (None, 28, 28, 32)        64
_________________________________________________________________
dense_64 (Dense)             (None, 28, 28, 64)        2112
_________________________________________________________________
flatten_21 (Flatten)         (None, 50176)             0
_________________________________________________________________
dense_65 (Dense)             (None, 128)               6422656
_________________________________________________________________
dropout_20 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_66 (Dense)             (None, 10)                1290
=================================================================
Total params: 6,426,122
Trainable params: 6,426,122
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
439/439 [==============================] - 6s 13ms/step - loss: 2.5641 -
accuracy: 0.5516 - precision: 0.7483 - recall: 0.4283 - val_loss: 0.4579
- val_accuracy: 0.8913 - val_precision: 0.9146 - val_recall: 0.8734
Epoch 2/100
439/439 [==============================] - 5s 12ms/step - loss: 0.6291 -
accuracy: 0.8000 - precision: 0.8749 - recall: 0.7328 - val_loss: 0.2885
- val_accuracy: 0.9248 - val_precision: 0.9536 - val_recall: 0.8866
Epoch 3/100
439/439 [==============================] - 5s 12ms/step - loss: 0.5281 -
accuracy: 0.8320 - precision: 0.8874 - recall: 0.7802 - val_loss: 0.2769
- val_accuracy: 0.9264 - val_precision: 0.9468 - val_recall: 0.9092
Epoch 4/100
439/439 [==============================] - 5s 12ms/step - loss: 0.5307 -
accuracy: 0.8348 - precision: 0.8777 - recall: 0.7971 - val_loss: 0.2822
- val_accuracy: 0.9258 - val_precision: 0.9504 - val_recall: 0.8993
Epoch 5/100
439/439 [==============================] - 5s 12ms/step - loss: 0.6051 -
accuracy: 0.8107 - precision: 0.8614 - recall: 0.7684 - val_loss: 0.2608
- val_accuracy: 0.9299 - val_precision: 0.9458 - val_recall: 0.9134
Epoch 6/100
439/439 [==============================] - 5s 12ms/step - loss: 0.4968 -
accuracy: 0.8454 - precision: 0.8837 - recall: 0.8143 - val_loss: 0.2928
- val_accuracy: 0.9188 - val_precision: 0.9445 - val_recall: 0.8965
Epoch 7/100
439/439 [==============================] - 5s 12ms/step - loss: 0.5996 -
accuracy: 0.8170 - precision: 0.8669 - recall: 0.7753 - val_loss: 0.3559
- val_accuracy: 0.9036 - val_precision: 0.9382 - val_recall: 0.8711
Epoch 8/100
439/439 [==============================] - 5s 12ms/step - loss: 0.5959 -
accuracy: 0.8130 - precision: 0.8761 - recall: 0.7515 - val_loss: 0.3750
- val_accuracy: 0.9023 - val_precision: 0.9415 - val_recall: 0.8524
Restoring model weights from the end of the best epoch.
Epoch 00008: early stopping
```

```
Test loss: 0.28543806076049805
Test accuracy: 0.9227142930030823
Test precision: 0.942704439163208
Test recall: 0.9072856903076172
Test f1 score: 0.9227142857142857
Test AUC for digit: 0 0.9789563324393538
Test AUC for digit: 1 0.979840545957394
Test AUC for digit: 2 0.9702686971098221
Test AUC for digit: 3 0.927925323986961
Test AUC for digit: 4 0.9539853844616494
Test AUC for digit: 5 0.9262426101687231
Test AUC for digit: 6 0.9711111111111111
Test AUC for digit: 7 0.960725329011793
Test AUC for digit: 8 0.9488095238095239
Test AUC for digit: 9 0.9511993241314747
```





We took our first model and decided to train it with different learning rates. With learning rate 0.05 we received very bad results (accuracy around 92%). The scores are so bad because our optimizer did not find good weights, because it had to change values with too big "jump".

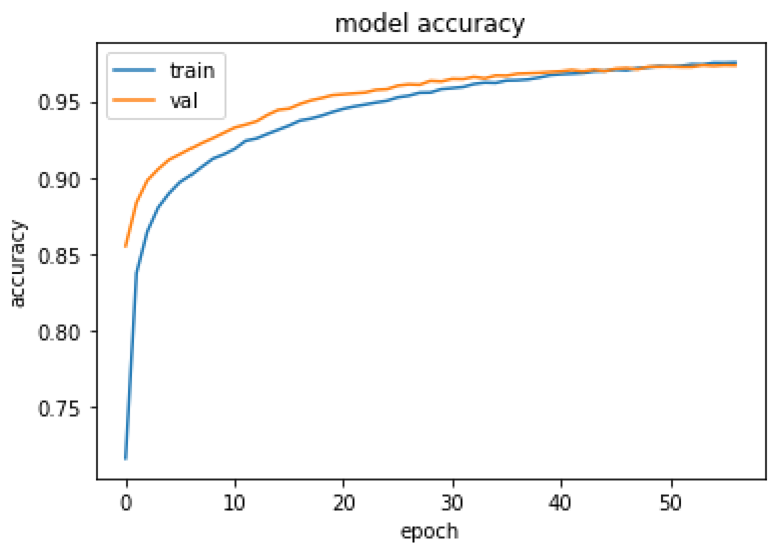
In [39]:
```
model_fc_00001 = keras.Sequential([
    layers.Dense(32, activation="relu",input_shape=(28,28,1)),
    layers.Dense(64, activation="relu"),

    layers.Flatten(),
    layers.Dense(128, activation="relu"),
    layers.Dropout(.25),
    layers.Dense(10, activation="softmax")
])
model_fc_00001.summary()
predict_model(model_fc_00001,[es], epochs=100, lr = 0.00001)
```

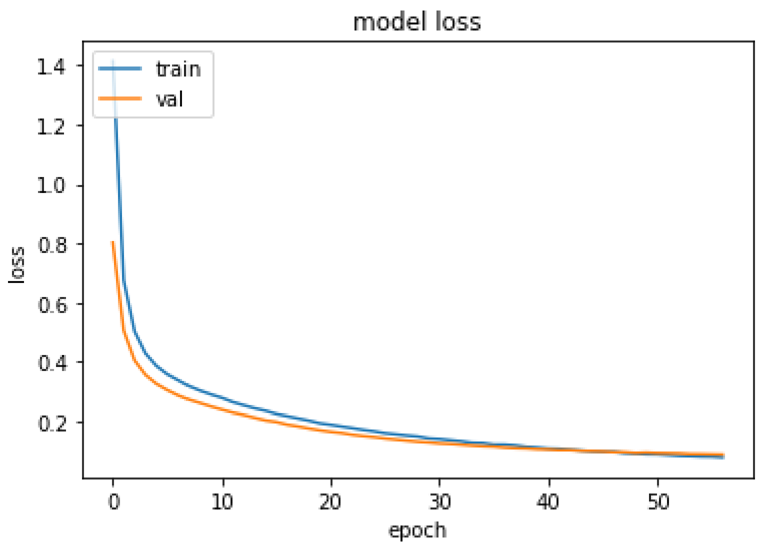
```
Model: "sequential_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_46 (Dense)             (None, 28, 28, 32)        64        
_________________________________________________________________
dense_47 (Dense)             (None, 28, 28, 64)        2112      
_________________________________________________________________
flatten_15 (Flatten)         (None, 50176)             0         
_________________________________________________________________
dense_48 (Dense)             (None, 128)               6422656   
_________________________________________________________________
dropout_14 (Dropout)         (None, 128)               0         
_________________________________________________________________
dense_49 (Dense)             (None, 10)                1290      
=================================================================
Total params: 6,426,122
Trainable params: 6,426,122
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
439/439 [==============================] - 6s 13ms/step - loss: 1.8177 -
accuracy: 0.5851 - precision: 0.7578 - recall: 0.0475 - val_loss: 0.8030
- val_accuracy: 0.8551 - val_precision: 0.9845 - val_recall: 0.5235
Epoch 2/100
439/439 [==============================] - 5s 12ms/step - loss: 0.7526 -
accuracy: 0.8255 - precision: 0.9614 - recall: 0.5864 - val_loss: 0.5059
- val_accuracy: 0.8835 - val_precision: 0.9580 - val_recall: 0.7645
Epoch 3/100
439/439 [==============================] - 5s 12ms/step - loss: 0.5323 -
accuracy: 0.8581 - precision: 0.9378 - recall: 0.7558 - val_loss: 0.4048
- val_accuracy: 0.8981 - val_precision: 0.9491 - val_recall: 0.8320
Epoch 4/100
439/439 [==============================] - 5s 12ms/step - loss: 0.4437 -
accuracy: 0.8764 - precision: 0.9324 - recall: 0.8142 - val_loss: 0.3561
- val_accuracy: 0.9055 - val_precision: 0.9460 - val_recall: 0.8603
Epoch 5/100
439/439 [==============================] - 5s 12ms/step - loss: 0.3928 -
accuracy: 0.8891 - precision: 0.9336 - recall: 0.8411 - val_loss: 0.3259
- val_accuracy: 0.9117 - val_precision: 0.9436 - val_recall: 0.8758
Epoch 6/100
439/439 [==============================] - 5s 12ms/step - loss: 0.3637 -
accuracy: 0.8953 - precision: 0.9334 - recall: 0.8587 - val_loss: 0.3050
- val_accuracy: 0.9153 - val_precision: 0.9434 - val_recall: 0.8876
Epoch 7/100
439/439 [==============================] - 5s 12ms/step - loss: 0.3479 -
accuracy: 0.8986 - precision: 0.9315 - recall: 0.8666 - val_loss: 0.2869
- val_accuracy: 0.9190 - val_precision: 0.9459 - val_recall: 0.8964
Epoch 8/100
439/439 [==============================] - 5s 12ms/step - loss: 0.3248 -
accuracy: 0.9051 - precision: 0.9349 - recall: 0.8769 - val_loss: 0.2723
- val_accuracy: 0.9227 - val_precision: 0.9477 - val_recall: 0.9000
Epoch 9/100
439/439 [==============================] - 5s 12ms/step - loss: 0.3000 -
accuracy: 0.9150 - precision: 0.9403 - recall: 0.8894 - val_loss: 0.2609
- val_accuracy: 0.9258 - val_precision: 0.9487 - val_recall: 0.9040
Epoch 10/100
439/439 [==============================] - 5s 12ms/step - loss: 0.2921 -
accuracy: 0.9155 - precision: 0.9409 - recall: 0.8923 - val_loss: 0.2493
- val_accuracy: 0.9293 - val_precision: 0.9492 - val_recall: 0.9094
Epoch 11/100
439/439 [==============================] - 5s 12ms/step - loss: 0.2765 -
```

```
accuracy: 0.9198 - precision: 0.9424 - recall: 0.8979 - val_loss: 0.2389
- val_accuracy: 0.9328 - val_precision: 0.9516 - val_recall: 0.9134
Epoch 12/100
439/439 [==============================] - 5s 12ms/step - loss: 0.2636 -
accuracy: 0.9249 - precision: 0.9456 - recall: 0.9054 - val_loss: 0.2288
- val_accuracy: 0.9346 - val_precision: 0.9539 - val_recall: 0.9189
Epoch 13/100
439/439 [==============================] - 5s 12ms/step - loss: 0.2534 -
accuracy: 0.9258 - precision: 0.9459 - recall: 0.9079 - val_loss: 0.2207
- val_accuracy: 0.9368 - val_precision: 0.9539 - val_recall: 0.9216
Epoch 14/100
439/439 [==============================] - 5s 12ms/step - loss: 0.2447 -
accuracy: 0.9276 - precision: 0.9473 - recall: 0.9091 - val_loss: 0.2116
- val_accuracy: 0.9410 - val_precision: 0.9564 - val_recall: 0.9264
Epoch 15/100
439/439 [==============================] - 5s 12ms/step - loss: 0.2372 -
accuracy: 0.9315 - precision: 0.9506 - recall: 0.9155 - val_loss: 0.2028
- val_accuracy: 0.9443 - val_precision: 0.9583 - val_recall: 0.9296
Epoch 16/100
439/439 [==============================] - 5s 12ms/step - loss: 0.2272 -
accuracy: 0.9334 - precision: 0.9515 - recall: 0.9177 - val_loss: 0.1975
- val_accuracy: 0.9452 - val_precision: 0.9590 - val_recall: 0.9326
Epoch 17/100
439/439 [==============================] - 5s 12ms/step - loss: 0.2178 -
accuracy: 0.9368 - precision: 0.9532 - recall: 0.9223 - val_loss: 0.1886
- val_accuracy: 0.9482 - val_precision: 0.9601 - val_recall: 0.9369
Epoch 18/100
439/439 [==============================] - 5s 12ms/step - loss: 0.2111 -
accuracy: 0.9389 - precision: 0.9525 - recall: 0.9257 - val_loss: 0.1827
- val_accuracy: 0.9505 - val_precision: 0.9626 - val_recall: 0.9390
Epoch 19/100
439/439 [==============================] - 5s 12ms/step - loss: 0.2045 -
accuracy: 0.9398 - precision: 0.9540 - recall: 0.9263 - val_loss: 0.1758
- val_accuracy: 0.9521 - val_precision: 0.9638 - val_recall: 0.9418
Epoch 20/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1990 -
accuracy: 0.9418 - precision: 0.9564 - recall: 0.9299 - val_loss: 0.1696
- val_accuracy: 0.9540 - val_precision: 0.9645 - val_recall: 0.9443
Epoch 21/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1892 -
accuracy: 0.9436 - precision: 0.9570 - recall: 0.9323 - val_loss: 0.1645
- val_accuracy: 0.9545 - val_precision: 0.9649 - val_recall: 0.9455
Epoch 22/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1896 -
accuracy: 0.9450 - precision: 0.9588 - recall: 0.9329 - val_loss: 0.1605
- val_accuracy: 0.9551 - val_precision: 0.9655 - val_recall: 0.9479
Epoch 23/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1779 -
accuracy: 0.9476 - precision: 0.9589 - recall: 0.9375 - val_loss: 0.1546
- val_accuracy: 0.9557 - val_precision: 0.9679 - val_recall: 0.9489
Epoch 24/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1749 -
accuracy: 0.9487 - precision: 0.9613 - recall: 0.9386 - val_loss: 0.1504
- val_accuracy: 0.9574 - val_precision: 0.9670 - val_recall: 0.9509
Epoch 25/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1666 -
accuracy: 0.9498 - precision: 0.9622 - recall: 0.9410 - val_loss: 0.1468
- val_accuracy: 0.9579 - val_precision: 0.9681 - val_recall: 0.9509
Epoch 26/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1642 -
accuracy: 0.9515 - precision: 0.9622 - recall: 0.9420 - val_loss: 0.1422
- val_accuracy: 0.9602 - val_precision: 0.9699 - val_recall: 0.9534
Epoch 27/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1584 -
accuracy: 0.9539 - precision: 0.9639 - recall: 0.9446 - val_loss: 0.1384
- val_accuracy: 0.9610 - val_precision: 0.9695 - val_recall: 0.9548
Epoch 28/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1521 -
```

```
accuracy: 0.9556 - precision: 0.9661 - recall: 0.9471 - val_loss: 0.1349
- val_accuracy: 0.9608 - val_precision: 0.9707 - val_recall: 0.9545
Epoch 29/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1494 -
accuracy: 0.9548 - precision: 0.9648 - recall: 0.9473 - val_loss: 0.1314
- val_accuracy: 0.9633 - val_precision: 0.9719 - val_recall: 0.9567
Epoch 30/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1435 -
accuracy: 0.9581 - precision: 0.9669 - recall: 0.9507 - val_loss: 0.1290
- val_accuracy: 0.9629 - val_precision: 0.9714 - val_recall: 0.9569
Epoch 31/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1403 -
accuracy: 0.9578 - precision: 0.9669 - recall: 0.9508 - val_loss: 0.1257
- val_accuracy: 0.9646 - val_precision: 0.9718 - val_recall: 0.9586
Epoch 32/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1395 -
accuracy: 0.9596 - precision: 0.9680 - recall: 0.9526 - val_loss: 0.1241
- val_accuracy: 0.9645 - val_precision: 0.9722 - val_recall: 0.9595
Epoch 33/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1311 -
accuracy: 0.9620 - precision: 0.9695 - recall: 0.9547 - val_loss: 0.1213
- val_accuracy: 0.9659 - val_precision: 0.9721 - val_recall: 0.9609
Epoch 34/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1297 -
accuracy: 0.9622 - precision: 0.9695 - recall: 0.9555 - val_loss: 0.1186
- val_accuracy: 0.9649 - val_precision: 0.9721 - val_recall: 0.9605
Epoch 35/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1279 -
accuracy: 0.9621 - precision: 0.9696 - recall: 0.9551 - val_loss: 0.1162
- val_accuracy: 0.9668 - val_precision: 0.9726 - val_recall: 0.9629
Epoch 36/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1146 -
accuracy: 0.9653 - precision: 0.9719 - recall: 0.9593 - val_loss: 0.1150
- val_accuracy: 0.9667 - val_precision: 0.9724 - val_recall: 0.9616
Epoch 37/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1249 -
accuracy: 0.9629 - precision: 0.9696 - recall: 0.9567 - val_loss: 0.1117
- val_accuracy: 0.9680 - val_precision: 0.9730 - val_recall: 0.9629
Epoch 38/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1214 -
accuracy: 0.9626 - precision: 0.9702 - recall: 0.9565 - val_loss: 0.1097
- val_accuracy: 0.9683 - val_precision: 0.9736 - val_recall: 0.9635
Epoch 39/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1171 -
accuracy: 0.9645 - precision: 0.9716 - recall: 0.9592 - val_loss: 0.1087
- val_accuracy: 0.9685 - val_precision: 0.9730 - val_recall: 0.9638
Epoch 40/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1124 -
accuracy: 0.9678 - precision: 0.9735 - recall: 0.9615 - val_loss: 0.1062
- val_accuracy: 0.9691 - val_precision: 0.9735 - val_recall: 0.9649
Epoch 41/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1069 -
accuracy: 0.9684 - precision: 0.9746 - recall: 0.9629 - val_loss: 0.1053
- val_accuracy: 0.9696 - val_precision: 0.9739 - val_recall: 0.9651
Epoch 42/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1087 -
accuracy: 0.9674 - precision: 0.9734 - recall: 0.9625 - val_loss: 0.1033
- val_accuracy: 0.9704 - val_precision: 0.9747 - val_recall: 0.9657
Epoch 43/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1096 -
accuracy: 0.9669 - precision: 0.9732 - recall: 0.9618 - val_loss: 0.1032
- val_accuracy: 0.9696 - val_precision: 0.9744 - val_recall: 0.9655
Epoch 44/100
439/439 [==============================] - 5s 12ms/step - loss: 0.1057 -
accuracy: 0.9685 - precision: 0.9743 - recall: 0.9640 - val_loss: 0.1008
- val_accuracy: 0.9707 - val_precision: 0.9743 - val_recall: 0.9671
Epoch 45/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0991 -
```

```
accuracy: 0.9693 - precision: 0.9747 - recall: 0.9651 - val_loss: 0.0997
- val_accuracy: 0.9697 - val_precision: 0.9745 - val_recall: 0.9668
Epoch 46/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0953 -
accuracy: 0.9718 - precision: 0.9766 - recall: 0.9676 - val_loss: 0.0985
- val_accuracy: 0.9713 - val_precision: 0.9746 - val_recall: 0.9681
Epoch 47/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0971 -
accuracy: 0.9703 - precision: 0.9753 - recall: 0.9661 - val_loss: 0.0962
- val_accuracy: 0.9717 - val_precision: 0.9759 - val_recall: 0.9696
Epoch 48/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0926 -
accuracy: 0.9709 - precision: 0.9765 - recall: 0.9662 - val_loss: 0.0961
- val_accuracy: 0.9709 - val_precision: 0.9753 - val_recall: 0.9684
Epoch 49/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0916 -
accuracy: 0.9721 - precision: 0.9774 - recall: 0.9678 - val_loss: 0.0933
- val_accuracy: 0.9726 - val_precision: 0.9769 - val_recall: 0.9703
Epoch 50/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0899 -
accuracy: 0.9725 - precision: 0.9768 - recall: 0.9677 - val_loss: 0.0946
- val_accuracy: 0.9726 - val_precision: 0.9762 - val_recall: 0.9688
Epoch 51/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0869 -
accuracy: 0.9727 - precision: 0.9777 - recall: 0.9684 - val_loss: 0.0929
- val_accuracy: 0.9727 - val_precision: 0.9762 - val_recall: 0.9701
Epoch 52/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0876 -
accuracy: 0.9723 - precision: 0.9773 - recall: 0.9680 - val_loss: 0.0924
- val_accuracy: 0.9724 - val_precision: 0.9763 - val_recall: 0.9700
Epoch 53/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0830 -
accuracy: 0.9746 - precision: 0.9792 - recall: 0.9706 - val_loss: 0.0919
- val_accuracy: 0.9724 - val_precision: 0.9765 - val_recall: 0.9706
Epoch 54/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0827 -
accuracy: 0.9737 - precision: 0.9790 - recall: 0.9706 - val_loss: 0.0897
- val_accuracy: 0.9736 - val_precision: 0.9774 - val_recall: 0.9716
Epoch 55/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0810 -
accuracy: 0.9750 - precision: 0.9791 - recall: 0.9705 - val_loss: 0.0894
- val_accuracy: 0.9732 - val_precision: 0.9768 - val_recall: 0.9716
Epoch 56/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0818 -
accuracy: 0.9751 - precision: 0.9793 - recall: 0.9714 - val_loss: 0.0887
- val_accuracy: 0.9736 - val_precision: 0.9769 - val_recall: 0.9711
Epoch 57/100
439/439 [==============================] - 5s 12ms/step - loss: 0.0783 -
accuracy: 0.9754 - precision: 0.9790 - recall: 0.9723 - val_loss: 0.0880
- val_accuracy: 0.9734 - val_precision: 0.9772 - val_recall: 0.9713
Epoch 00057: early stopping
Test loss: 0.10033061355352402
Test accuracy: 0.9710000157356262
Test precision: 0.9738430380821228
Test recall: 0.9679999947547913
Test f1 score: 0.971
Test AUC for digit: 0 0.9930644081388708
Test AUC for digit: 1 0.9911542498484037
Test AUC for digit: 2 0.9863290741003511
Test AUC for digit: 3 0.9797077491850635
Test AUC for digit: 4 0.986564103792539
Test AUC for digit: 5 0.9774254759105543
Test AUC for digit: 6 0.9903174603174603
Test AUC for digit: 7 0.9857344204244699
Test AUC for digit: 8 0.9765873015873017
Test AUC for digit: 9 0.9711964862665278
```





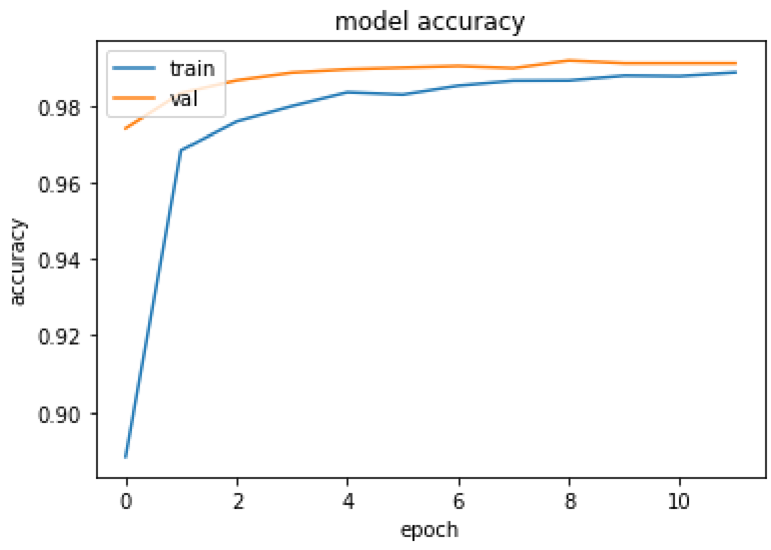
Model with learning rate equals 0.00001 performed pretty well but it needed 54 epochs to achieve 97,1% accuracy (compared to 6 epochs using standard learning rate equals 0.001). This is because optimizer "jumped" too small distance searching best results, and it had to do many iterations to find the best weights.

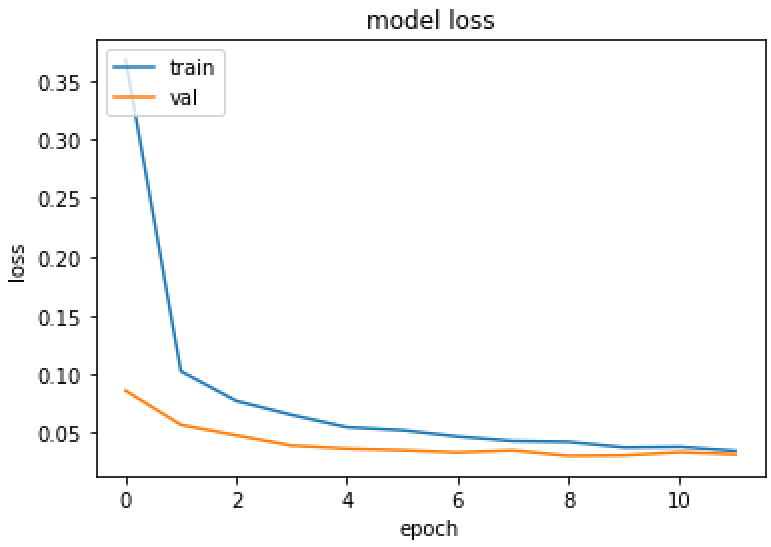
Basic Multi-layer CNN

In [17]:
```
model_cnn = keras.Sequential([
    layers.Conv2D(32, (3,3), activation="relu", input_shape=(28,28,1)),
    layers.MaxPooling2D (2,2),
    layers.Conv2D(64, (3,3), activation="relu"),
    layers.MaxPooling2D (2,2),

    layers.Flatten(),
    layers.Dropout(.5),
    layers.Dense(10, activation="softmax")
])
model_cnn.summary()
predict_model(model_cnn, [es], epochs=100)
```

```
Model: "sequential_5"
_________________________________________________________________
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, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
flatten_5 (Flatten)          (None, 1600)              0
_________________________________________________________________
dropout_5 (Dropout)          (None, 1600)              0
_________________________________________________________________
dense_21 (Dense)             (None, 10)                16010
=================================================================
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
439/439 [==============================] - 22s 28ms/step - loss: 0.7757 -
accuracy: 0.7586 - precision: 0.8819 - recall: 0.6326 - val_loss: 0.0861
- val_accuracy: 0.9740 - val_precision: 0.9813 - val_recall: 0.9697
Epoch 2/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1093 - a
ccuracy: 0.9654 - precision: 0.9719 - recall: 0.9608 - val_loss: 0.0568 -
val_accuracy: 0.9833 - val_precision: 0.9858 - val_recall: 0.9804
Epoch 3/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0831 - a
ccuracy: 0.9750 - precision: 0.9788 - recall: 0.9721 - val_loss: 0.0478 -
val_accuracy: 0.9866 - val_precision: 0.9887 - val_recall: 0.9846
Epoch 4/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0692 - a
ccuracy: 0.9781 - precision: 0.9811 - recall: 0.9756 - val_loss: 0.0391 -
val_accuracy: 0.9886 - val_precision: 0.9910 - val_recall: 0.9869
Epoch 5/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0525 - a
ccuracy: 0.9844 - precision: 0.9866 - recall: 0.9822 - val_loss: 0.0364 -
val_accuracy: 0.9895 - val_precision: 0.9920 - val_recall: 0.9879
Epoch 6/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0548 - a
ccuracy: 0.9820 - precision: 0.9843 - recall: 0.9804 - val_loss: 0.0351 -
val_accuracy: 0.9899 - val_precision: 0.9918 - val_recall: 0.9890
Epoch 7/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0452 - a
ccuracy: 0.9856 - precision: 0.9872 - recall: 0.9844 - val_loss: 0.0332 -
val_accuracy: 0.9903 - val_precision: 0.9923 - val_recall: 0.9895
Epoch 8/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0416 - a
ccuracy: 0.9865 - precision: 0.9884 - recall: 0.9850 - val_loss: 0.0349 -
val_accuracy: 0.9898 - val_precision: 0.9910 - val_recall: 0.9892
Epoch 9/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0412 - a
ccuracy: 0.9868 - precision: 0.9884 - recall: 0.9854 - val_loss: 0.0303 -
val_accuracy: 0.9918 - val_precision: 0.9928 - val_recall: 0.9908
Epoch 10/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0366 - a
ccuracy: 0.9881 - precision: 0.9893 - recall: 0.9870 - val_loss: 0.0307 -
val_accuracy: 0.9911 - val_precision: 0.9919 - val_recall: 0.9899
Epoch 11/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0414 - a
ccuracy: 0.9866 - precision: 0.9877 - recall: 0.9853 - val_loss: 0.0335 -
val_accuracy: 0.9911 - val_precision: 0.9918 - val_recall: 0.9906
Epoch 12/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0345 - a
ccuracy: 0.9890 - precision: 0.9897 - recall: 0.9881 - val_loss: 0.0316 -
val_accuracy: 0.9911 - val_precision: 0.9922 - val_recall: 0.9903
Epoch 00012: early stopping
Test loss: 0.03780661150813103
Test accuracy: 0.9877142906188965
Test precision: 0.988834798336029
Test recall: 0.9868571162223816
Test f1 score: 0.9877142857142858
Test AUC for digit: 0 0.9986987490590519
Test AUC for digit: 1 0.9946251332301316
Test AUC for digit: 2 0.9912251490762208
Test AUC for digit: 3 0.9914630429597434
Test AUC for digit: 4 0.9935349833293058
Test AUC for digit: 5 0.9949048474533845
Test AUC for digit: 6 0.9943650793650793
Test AUC for digit: 7 0.9923824507574553
Test AUC for digit: 8 0.9917460317460318
Test AUC for digit: 9 0.9890579103854441
```





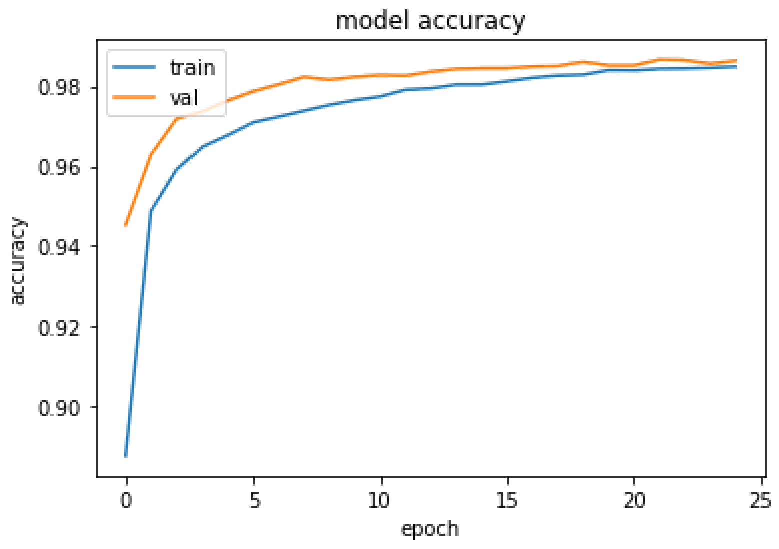
Our first convolutional model with 2 convolutional layers was performing even better then fully connected neural networks. This model is not overfitted, because train and validation loss and accuracy are close to each other. It has only 34,826 parameters to train, so the training of such model is pretty fast. On test set model achieves 98.7% accuracy which is great result.

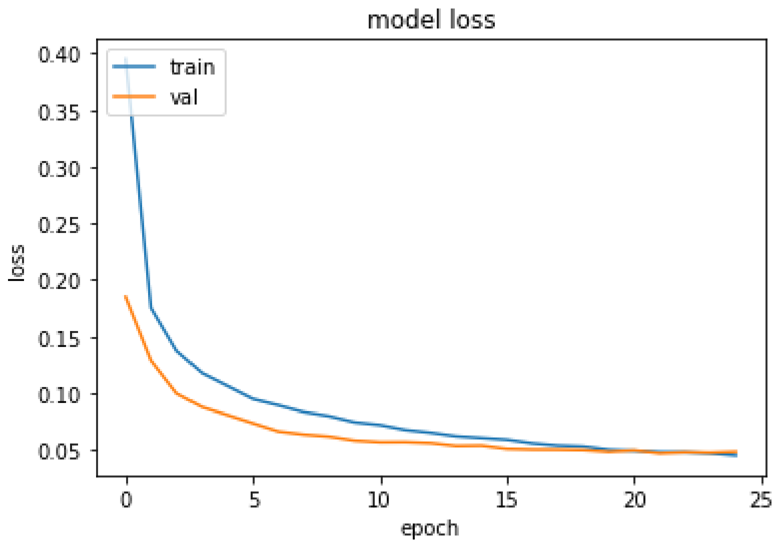
Different number of convolutional layers

In [18]:
```
model_cnn_short = keras.Sequential([
    layers.Conv2D(32, (3,3), activation="relu", input_shape=(28,28,1)),
    layers.MaxPooling2D (2,2),

    layers.Flatten(),
    layers.Dropout(.5),
    layers.Dense(10, activation="softmax")
])
model_cnn_short.summary()
predict_model(model_cnn_short, [es], epochs=100)
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_2 (Conv2D)            (None, 26, 26, 32)        320       
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 13, 13, 32)        0         
_________________________________________________________________
flatten_6 (Flatten)          (None, 5408)              0         
_________________________________________________________________
dropout_6 (Dropout)          (None, 5408)              0         
_________________________________________________________________
dense_22 (Dense)             (None, 10)                54090     
=================================================================
Total params: 54,410
Trainable params: 54,410
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
439/439 [==============================] - 4s 8ms/step - loss: 0.7126 - a
ccuracy: 0.7976 - precision: 0.9174 - recall: 0.6466 - val_loss: 0.1849 -
val_accuracy: 0.9453 - val_precision: 0.9596 - val_recall: 0.9362
Epoch 2/100
439/439 [==============================] - 3s 6ms/step - loss: 0.1896 - a
ccuracy: 0.9442 - precision: 0.9575 - recall: 0.9324 - val_loss: 0.1285 -
val_accuracy: 0.9629 - val_precision: 0.9724 - val_recall: 0.9567
Epoch 3/100
439/439 [==============================] - 3s 6ms/step - loss: 0.1387 - a
ccuracy: 0.9582 - precision: 0.9662 - recall: 0.9508 - val_loss: 0.0996 -
val_accuracy: 0.9719 - val_precision: 0.9783 - val_recall: 0.9678
Epoch 4/100
439/439 [==============================] - 3s 6ms/step - loss: 0.1196 - a
ccuracy: 0.9648 - precision: 0.9723 - recall: 0.9583 - val_loss: 0.0876 -
val_accuracy: 0.9737 - val_precision: 0.9791 - val_recall: 0.9713
Epoch 5/100
439/439 [==============================] - 3s 6ms/step - loss: 0.1106 - a
ccuracy: 0.9663 - precision: 0.9729 - recall: 0.9611 - val_loss: 0.0805 -
val_accuracy: 0.9765 - val_precision: 0.9801 - val_recall: 0.9739
Epoch 6/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0965 - a
ccuracy: 0.9702 - precision: 0.9750 - recall: 0.9659 - val_loss: 0.0732 -
val_accuracy: 0.9788 - val_precision: 0.9829 - val_recall: 0.9759
Epoch 7/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0908 - a
ccuracy: 0.9717 - precision: 0.9763 - recall: 0.9680 - val_loss: 0.0660 -
val_accuracy: 0.9805 - val_precision: 0.9827 - val_recall: 0.9771
Epoch 8/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0875 - a
ccuracy: 0.9728 - precision: 0.9768 - recall: 0.9696 - val_loss: 0.0633 -
val_accuracy: 0.9824 - val_precision: 0.9852 - val_recall: 0.9799
Epoch 9/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0787 - a
ccuracy: 0.9759 - precision: 0.9793 - recall: 0.9729 - val_loss: 0.0615 -
val_accuracy: 0.9817 - val_precision: 0.9856 - val_recall: 0.9789
Epoch 10/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0759 - a
ccuracy: 0.9759 - precision: 0.9799 - recall: 0.9731 - val_loss: 0.0580 -
val_accuracy: 0.9824 - val_precision: 0.9849 - val_recall: 0.9797
Epoch 11/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0730 - a
ccuracy: 0.9771 - precision: 0.9806 - recall: 0.9748 - val_loss: 0.0567 -
val_accuracy: 0.9828 - val_precision: 0.9849 - val_recall: 0.9798
```

```
Epoch 12/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0632 - a
ccuracy: 0.9803 - precision: 0.9833 - recall: 0.9784 - val_loss: 0.0567 -
val_accuracy: 0.9827 - val_precision: 0.9845 - val_recall: 0.9807
Epoch 13/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0649 - a
ccuracy: 0.9793 - precision: 0.9820 - recall: 0.9770 - val_loss: 0.0559 -
val_accuracy: 0.9837 - val_precision: 0.9854 - val_recall: 0.9820
Epoch 14/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0647 - a
ccuracy: 0.9799 - precision: 0.9823 - recall: 0.9780 - val_loss: 0.0534 -
val_accuracy: 0.9844 - val_precision: 0.9861 - val_recall: 0.9825
Epoch 15/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0569 - a
ccuracy: 0.9815 - precision: 0.9842 - recall: 0.9795 - val_loss: 0.0537 -
val_accuracy: 0.9846 - val_precision: 0.9857 - val_recall: 0.9834
Epoch 16/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0576 - a
ccuracy: 0.9815 - precision: 0.9833 - recall: 0.9797 - val_loss: 0.0508 -
val_accuracy: 0.9846 - val_precision: 0.9860 - val_recall: 0.9827
Epoch 17/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0528 - a
ccuracy: 0.9825 - precision: 0.9848 - recall: 0.9807 - val_loss: 0.0503 -
val_accuracy: 0.9850 - val_precision: 0.9864 - val_recall: 0.9843
Epoch 18/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0555 - a
ccuracy: 0.9822 - precision: 0.9844 - recall: 0.9808 - val_loss: 0.0502 -
val_accuracy: 0.9851 - val_precision: 0.9874 - val_recall: 0.9835
Epoch 19/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0516 - a
ccuracy: 0.9831 - precision: 0.9852 - recall: 0.9814 - val_loss: 0.0498 -
val_accuracy: 0.9861 - val_precision: 0.9877 - val_recall: 0.9848
Epoch 20/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0481 - a
ccuracy: 0.9842 - precision: 0.9860 - recall: 0.9827 - val_loss: 0.0483 -
val_accuracy: 0.9853 - val_precision: 0.9867 - val_recall: 0.9841
Epoch 21/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0466 - a
ccuracy: 0.9856 - precision: 0.9873 - recall: 0.9842 - val_loss: 0.0496 -
val_accuracy: 0.9853 - val_precision: 0.9870 - val_recall: 0.9843
Epoch 22/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0456 - a
ccuracy: 0.9849 - precision: 0.9869 - recall: 0.9834 - val_loss: 0.0469 -
val_accuracy: 0.9867 - val_precision: 0.9884 - val_recall: 0.9859
Epoch 23/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0454 - a
ccuracy: 0.9850 - precision: 0.9869 - recall: 0.9837 - val_loss: 0.0479 -
val_accuracy: 0.9866 - val_precision: 0.9876 - val_recall: 0.9848
Epoch 24/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0454 - a
ccuracy: 0.9855 - precision: 0.9874 - recall: 0.9840 - val_loss: 0.0476 -
val_accuracy: 0.9857 - val_precision: 0.9870 - val_recall: 0.9853
Epoch 25/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0442 - a
ccuracy: 0.9851 - precision: 0.9868 - recall: 0.9840 - val_loss: 0.0483 -
val_accuracy: 0.9864 - val_precision: 0.9871 - val_recall: 0.9853
Epoch 00025: early stopping
Test loss: 0.06184159591794014
Test accuracy: 0.9821428656578064
Test precision: 0.9840951561927795
Test recall: 0.9811428785324097
Test f1 score: 0.9821428571428571
Test AUC for digit: 0 0.9946403781193142
Test AUC for digit: 1 0.9928970988504665
Test AUC for digit: 2 0.9918050336696427
Test AUC for digit: 3 0.9915450335250312
Test AUC for digit: 4 0.9882236580836722
Test AUC for digit: 5 0.9855790503765007
Test AUC for digit: 6 0.9932539682539682
Test AUC for digit: 7 0.9851409580367276
Test AUC for digit: 8 0.9859523809523809
Test AUC for digit: 9 0.9908959987735204
```



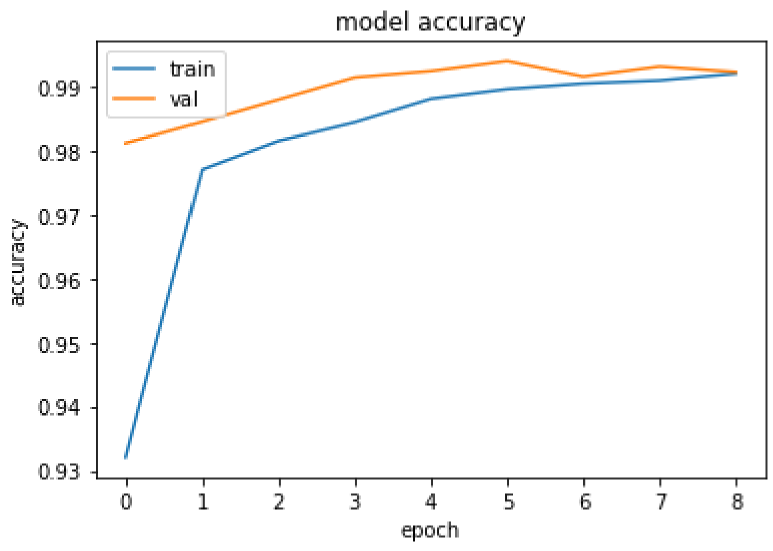


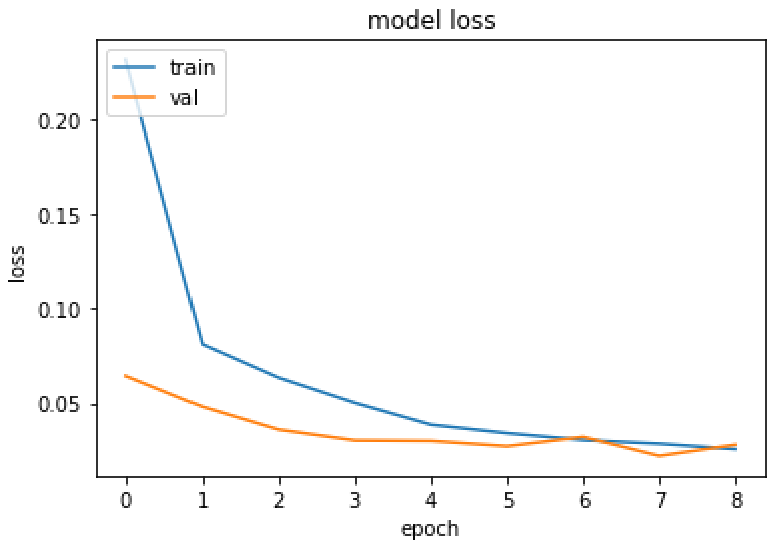
Next model has only 1 convolution layer which has more parameters (54,410) because of less number of pooling layers. The results are satisfying, but not as good as previous model (test accuracy equals 98,2%).

In [67]:
```
model_cnn_long = keras.Sequential([
    layers.Conv2D(32, (3,3), activation="relu", input_shape=(28,28,1)),
    layers.MaxPooling2D ((2,2),1),
    layers.Conv2D(64, (3,3), activation="relu"),
    layers.MaxPooling2D ((2,2),1),
    layers.Conv2D(128, (3,3), activation="relu"),
    layers.MaxPooling2D ((2,2),1),
    layers.Conv2D(512, (3,3), activation="relu"),
    layers.MaxPooling2D ((2,2),1),

    layers.Flatten(),
    layers.Dense(128, activation="relu"),
    layers.Dropout(.5),
    layers.Dense(10, activation="softmax")
])
model_cnn_long.summary()
predict_model(model_cnn_long, [es], epochs=100)
```

```
Model: "sequential_28"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_32 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 25, 25, 32)        0
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 23, 23, 64)        18496
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 22, 22, 64)        0
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 20, 20, 128)       73856
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 19, 19, 128)       0
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 17, 17, 512)       590336
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 16, 16, 512)       0
_________________________________________________________________
flatten_28 (Flatten)         (None, 131072)            0
_________________________________________________________________
dense_85 (Dense)             (None, 128)               16777344
_________________________________________________________________
dropout_22 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_86 (Dense)             (None, 10)                1290
=================================================================
Total params: 17,461,642
Trainable params: 17,461,642
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
439/439 [==============================] - 19s 42ms/step - loss: 0.5369 -
accuracy: 0.8446 - precision: 0.9042 - recall: 0.7958 - val_loss: 0.0644
- val_accuracy: 0.9812 - val_precision: 0.9828 - val_recall: 0.9805
Epoch 2/100
439/439 [==============================] - 18s 40ms/step - loss: 0.0900 -
accuracy: 0.9740 - precision: 0.9778 - recall: 0.9702 - val_loss: 0.0481
- val_accuracy: 0.9846 - val_precision: 0.9861 - val_recall: 0.9833
Epoch 3/100
439/439 [==============================] - 18s 40ms/step - loss: 0.0727 -
accuracy: 0.9790 - precision: 0.9818 - recall: 0.9766 - val_loss: 0.0360
- val_accuracy: 0.9880 - val_precision: 0.9900 - val_recall: 0.9864
Epoch 4/100
439/439 [==============================] - 18s 41ms/step - loss: 0.0553 -
accuracy: 0.9824 - precision: 0.9848 - recall: 0.9804 - val_loss: 0.0303
- val_accuracy: 0.9915 - val_precision: 0.9922 - val_recall: 0.9909
Epoch 5/100
439/439 [==============================] - 18s 40ms/step - loss: 0.0370 -
accuracy: 0.9888 - precision: 0.9900 - recall: 0.9874 - val_loss: 0.0300
- val_accuracy: 0.9925 - val_precision: 0.9935 - val_recall: 0.9922
Epoch 6/100
439/439 [==============================] - 18s 40ms/step - loss: 0.0322 -
accuracy: 0.9895 - precision: 0.9908 - recall: 0.9885 - val_loss: 0.0272
- val_accuracy: 0.9941 - val_precision: 0.9942 - val_recall: 0.9935
Epoch 7/100
439/439 [==============================] - 18s 41ms/step - loss: 0.0295 -
accuracy: 0.9911 - precision: 0.9922 - recall: 0.9904 - val_loss: 0.0321
- val_accuracy: 0.9916 - val_precision: 0.9923 - val_recall: 0.9915
Epoch 8/100
439/439 [==============================] - 18s 41ms/step - loss: 0.0277 -
accuracy: 0.9911 - precision: 0.9924 - recall: 0.9905 - val_loss: 0.0221
- val_accuracy: 0.9932 - val_precision: 0.9941 - val_recall: 0.9931
Epoch 9/100
439/439 [==============================] - 18s 41ms/step - loss: 0.0235 -
accuracy: 0.9925 - precision: 0.9931 - recall: 0.9920 - val_loss: 0.0280
- val_accuracy: 0.9924 - val_precision: 0.9925 - val_recall: 0.9924
```

```
Restoring model weights from the end of the best epoch.
Epoch 00009: early stopping
Test loss: 0.03389483317732811
Test accuracy: 0.9918571710586548
Test precision: 0.992139458656311
Test recall: 0.9917142987251282
Test f1 score: 0.9918571428571429
Test AUC for digit: 0 0.9990376444107582
Test AUC for digit: 1 0.9981951123812957
Test AUC for digit: 2 0.9929911952983329
Test AUC for digit: 3 0.9945313397184135
Test AUC for digit: 4 0.9947926711881807
Test AUC for digit: 5 0.9956330341254677
Test AUC for digit: 6 0.9968966508108128
Test AUC for digit: 7 0.999361124421019
Test AUC for digit: 8 0.9925846524567535
Test AUC for digit: 9 0.990134076231021
```





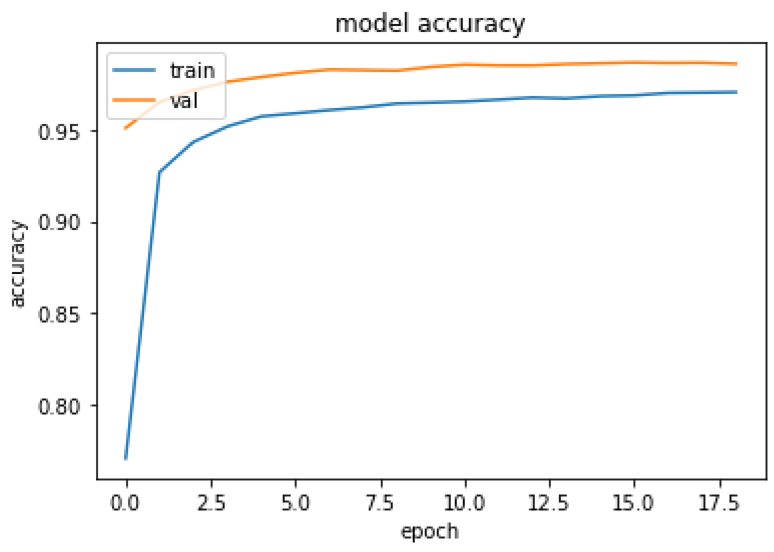
Next we created a neural network with 4 convolutional layers and with 17 milion parameters. The model was not overfitted. It had accuracy around 99.2% for test, train and validation model. Time needed to train this model was much higher (19s per epoch comparing to 3s per epoch in CNN that we have implemented). This is the best model that we have created for this dataset.

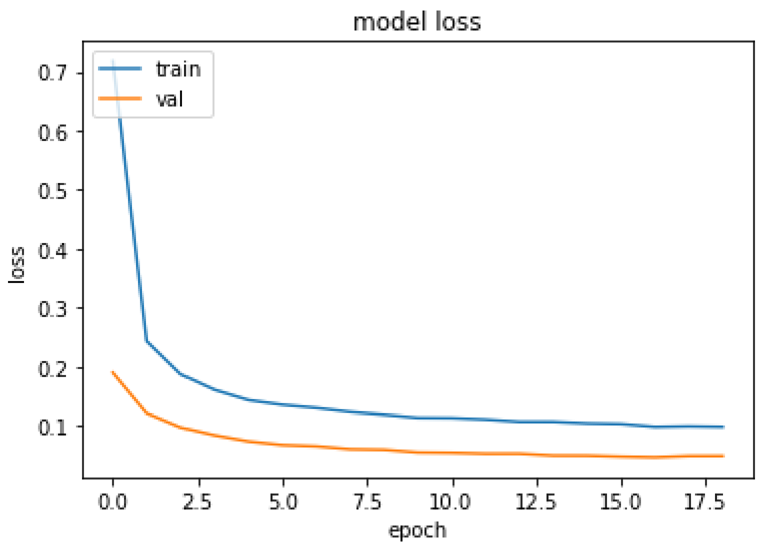
Different number of filters per layer

In [21]:
```
model_cnn_min = keras.Sequential([
    layers.Conv2D(4, (3,3), activation="relu", input_shape=(28,28,1)),
    layers.MaxPooling2D (2,2),
    layers.Conv2D(16, (3,3), activation="relu"),
    layers.MaxPooling2D (2,2),

    layers.Flatten(),
    layers.Dropout(.5),
    layers.Dense(10, activation="softmax")
])
model_cnn_min.summary()
predict_model(model_cnn_min, [es], epochs=100)
```

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_9 (Conv2D)            (None, 26, 26, 4)         40        
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 13, 13, 4)         0         
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 11, 11, 16)        592       
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 5, 5, 16)          0         
_________________________________________________________________
flatten_9 (Flatten)          (None, 400)               0         
_________________________________________________________________
dropout_8 (Dropout)          (None, 400)               0         
_________________________________________________________________
dense_28 (Dense)             (None, 10)                4010      
=================================================================
Total params: 4,642
Trainable params: 4,642
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
439/439 [==============================] - 20s 29ms/step - loss: 1.2412 -
accuracy: 0.5863 - precision: 0.8270 - recall: 0.3904 - val_loss: 0.1895
- val_accuracy: 0.9508 - val_precision: 0.9644 - val_recall: 0.9309
Epoch 2/100
439/439 [==============================] - 3s 6ms/step - loss: 0.2660 - a
ccuracy: 0.9202 - precision: 0.9399 - recall: 0.9013 - val_loss: 0.1206 -
val_accuracy: 0.9649 - val_precision: 0.9742 - val_recall: 0.9571
Epoch 3/100
439/439 [==============================] - 3s 6ms/step - loss: 0.1972 - a
ccuracy: 0.9398 - precision: 0.9519 - recall: 0.9284 - val_loss: 0.0964 -
val_accuracy: 0.9716 - val_precision: 0.9768 - val_recall: 0.9651
Epoch 4/100
439/439 [==============================] - 3s 6ms/step - loss: 0.1631 - a
ccuracy: 0.9512 - precision: 0.9599 - recall: 0.9423 - val_loss: 0.0831 -
val_accuracy: 0.9760 - val_precision: 0.9807 - val_recall: 0.9701
Epoch 5/100
439/439 [==============================] - 3s 6ms/step - loss: 0.1448 - a
ccuracy: 0.9565 - precision: 0.9640 - recall: 0.9497 - val_loss: 0.0728 -
val_accuracy: 0.9786 - val_precision: 0.9820 - val_recall: 0.9747
Epoch 6/100
439/439 [==============================] - 3s 6ms/step - loss: 0.1358 - a
ccuracy: 0.9586 - precision: 0.9660 - recall: 0.9528 - val_loss: 0.0668 -
val_accuracy: 0.9810 - val_precision: 0.9846 - val_recall: 0.9784
Epoch 7/100
439/439 [==============================] - 3s 6ms/step - loss: 0.1321 - a
ccuracy: 0.9604 - precision: 0.9674 - recall: 0.9548 - val_loss: 0.0647 -
val_accuracy: 0.9827 - val_precision: 0.9860 - val_recall: 0.9788
Epoch 8/100
439/439 [==============================] - 3s 6ms/step - loss: 0.1219 - a
ccuracy: 0.9629 - precision: 0.9683 - recall: 0.9573 - val_loss: 0.0597 -
val_accuracy: 0.9824 - val_precision: 0.9862 - val_recall: 0.9801
Epoch 9/100
```

```
439/439 [==============================] - 3s 6ms/step - loss: 0.1195 - a
ccuracy: 0.9628 - precision: 0.9687 - recall: 0.9579 - val_loss: 0.0588 -
val_accuracy: 0.9821 - val_precision: 0.9861 - val_recall: 0.9805
Epoch 10/100
439/439 [==============================] - 3s 6ms/step - loss: 0.1119 - a
ccuracy: 0.9654 - precision: 0.9695 - recall: 0.9610 - val_loss: 0.0542 -
val_accuracy: 0.9841 - val_precision: 0.9865 - val_recall: 0.9818
Epoch 11/100
439/439 [==============================] - 3s 6ms/step - loss: 0.1154 - a
ccuracy: 0.9644 - precision: 0.9700 - recall: 0.9598 - val_loss: 0.0537 -
val_accuracy: 0.9854 - val_precision: 0.9875 - val_recall: 0.9827
Epoch 12/100
439/439 [==============================] - 3s 6ms/step - loss: 0.1106 - a
ccuracy: 0.9667 - precision: 0.9717 - recall: 0.9622 - val_loss: 0.0522 -
val_accuracy: 0.9850 - val_precision: 0.9875 - val_recall: 0.9827
Epoch 13/100
439/439 [==============================] - 3s 6ms/step - loss: 0.1048 - a
ccuracy: 0.9677 - precision: 0.9727 - recall: 0.9641 - val_loss: 0.0522 -
val_accuracy: 0.9850 - val_precision: 0.9868 - val_recall: 0.9830
Epoch 14/100
439/439 [==============================] - 3s 6ms/step - loss: 0.1069 - a
ccuracy: 0.9672 - precision: 0.9715 - recall: 0.9638 - val_loss: 0.0490 -
val_accuracy: 0.9857 - val_precision: 0.9877 - val_recall: 0.9843
Epoch 15/100
439/439 [==============================] - 3s 6ms/step - loss: 0.1051 - a
ccuracy: 0.9676 - precision: 0.9712 - recall: 0.9637 - val_loss: 0.0488 -
val_accuracy: 0.9861 - val_precision: 0.9875 - val_recall: 0.9835
Epoch 16/100
439/439 [==============================] - 3s 6ms/step - loss: 0.1029 - a
ccuracy: 0.9683 - precision: 0.9719 - recall: 0.9650 - val_loss: 0.0472 -
val_accuracy: 0.9866 - val_precision: 0.9884 - val_recall: 0.9851
Epoch 17/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0990 - a
ccuracy: 0.9694 - precision: 0.9730 - recall: 0.9657 - val_loss: 0.0461 -
val_accuracy: 0.9863 - val_precision: 0.9883 - val_recall: 0.9847
Epoch 18/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0960 - a
ccuracy: 0.9705 - precision: 0.9743 - recall: 0.9670 - val_loss: 0.0482 -
val_accuracy: 0.9864 - val_precision: 0.9881 - val_recall: 0.9838
Epoch 19/100
439/439 [==============================] - 3s 6ms/step - loss: 0.0978 - a
ccuracy: 0.9701 - precision: 0.9744 - recall: 0.9665 - val_loss: 0.0482 -
val_accuracy: 0.9859 - val_precision: 0.9877 - val_recall: 0.9840
Epoch 00019: early stopping
Test loss: 0.06978869438171387
Test accuracy: 0.9787142872810364
Test precision: 0.9813513159751892
Test recall: 0.9772857427597046
Test f1 score: 0.9787142857142858
Test AUC for digit: 0 0.9965322040694353
Test AUC for digit: 1 0.9950131545729289
Test AUC for digit: 2 0.9902744281128235
Test AUC for digit: 3 0.9815570919354413
Test AUC for digit: 4 0.9918910464386338
Test AUC for digit: 5 0.9894167367175205
Test AUC for digit: 6 0.9894444444444445
Test AUC for digit: 7 0.9815003368863819
Test AUC for digit: 8 0.9804761904761905
Test AUC for digit: 9 0.9860031281752693
```



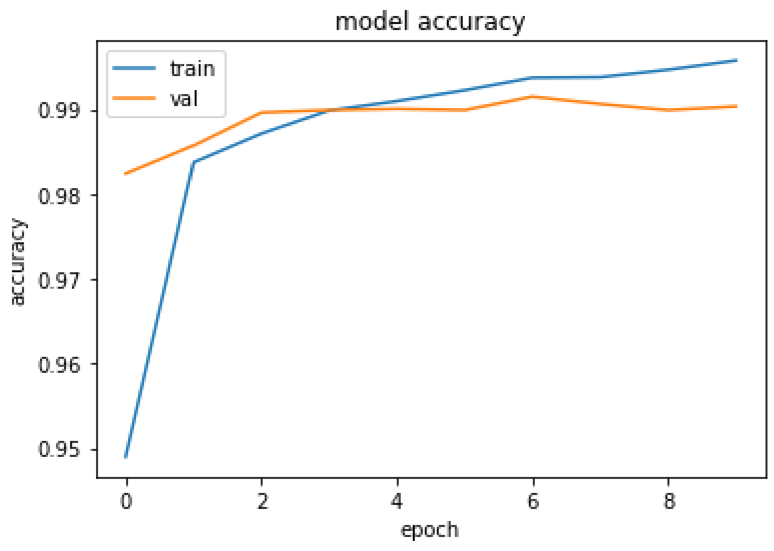


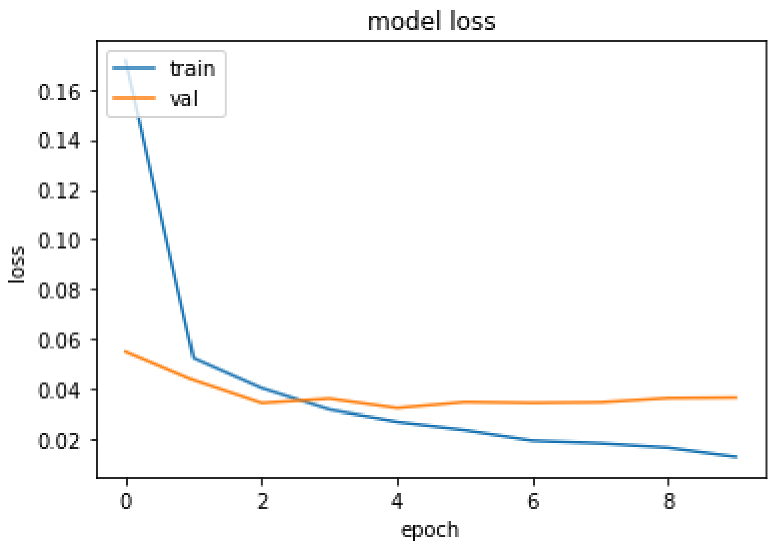
Next we decided to check how number of filters impact to performance of model. Reducing number of filter in convolutional layers made our model worse than basic model. Accuracy has fallen to 97.8%, because this model was too simple to explain complexity of our data. This model is underfitted.

In [22]:
```
model_cnn_max = keras.Sequential([
    layers.Conv2D(128, (3,3), activation="relu", input_shape=(28,28,1)),
    layers.MaxPooling2D (2,2),
    layers.Conv2D(512, (3,3), activation="relu"),
    layers.MaxPooling2D (2,2),

    layers.Flatten(),
    layers.Dropout(.5),
    layers.Dense(10, activation="softmax")
])
model_cnn_max.summary()
predict_model(model_cnn_max, [es], epochs=100)
```

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_11 (Conv2D)           (None, 26, 26, 128)       1280
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 13, 13, 128)       0
_________________________________________________________________
```

```
conv2d_12 (Conv2D)           (None, 11, 11, 512)       590336    
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 5, 5, 512)         0         
_________________________________________________________________
flatten_10 (Flatten)         (None, 12800)             0         
_________________________________________________________________
dropout_9 (Dropout)          (None, 12800)             0         
_________________________________________________________________
dense_29 (Dense)             (None, 10)                128010    
=================================================================
Total params: 719,626
Trainable params: 719,626
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
439/439 [==============================] - 27s 44ms/step - loss: 0.3990 - 
accuracy: 0.8824 - precision: 0.9333 - recall: 0.8202 - val_loss: 0.0549 
- val_accuracy: 0.9824 - val_precision: 0.9854 - val_recall: 0.9804
Epoch 2/100
439/439 [==============================] - 9s 19ms/step - loss: 0.0521 - 
accuracy: 0.9837 - precision: 0.9856 - recall: 0.9818 - val_loss: 0.0435 
- val_accuracy: 0.9857 - val_precision: 0.9870 - val_recall: 0.9844
Epoch 3/100
439/439 [==============================] - 8s 19ms/step - loss: 0.0429 - 
accuracy: 0.9862 - precision: 0.9876 - recall: 0.9849 - val_loss: 0.0343 
- val_accuracy: 0.9896 - val_precision: 0.9902 - val_recall: 0.9885
Epoch 4/100
439/439 [==============================] - 8s 19ms/step - loss: 0.0320 - 
accuracy: 0.9897 - precision: 0.9904 - recall: 0.9890 - val_loss: 0.0361 
- val_accuracy: 0.9899 - val_precision: 0.9909 - val_recall: 0.9895
Epoch 5/100
439/439 [==============================] - 8s 19ms/step - loss: 0.0249 - 
accuracy: 0.9921 - precision: 0.9929 - recall: 0.9916 - val_loss: 0.0323 
- val_accuracy: 0.9900 - val_precision: 0.9913 - val_recall: 0.9890
Epoch 6/100
439/439 [==============================] - 8s 19ms/step - loss: 0.0233 - 
accuracy: 0.9925 - precision: 0.9928 - recall: 0.9920 - val_loss: 0.0347 
- val_accuracy: 0.9899 - val_precision: 0.9903 - val_recall: 0.9896
Epoch 7/100
439/439 [==============================] - 8s 19ms/step - loss: 0.0185 - 
accuracy: 0.9941 - precision: 0.9945 - recall: 0.9938 - val_loss: 0.0343 
- val_accuracy: 0.9915 - val_precision: 0.9921 - val_recall: 0.9911
Epoch 8/100
439/439 [==============================] - 8s 19ms/step - loss: 0.0156 - 
accuracy: 0.9946 - precision: 0.9950 - recall: 0.9943 - val_loss: 0.0346 
- val_accuracy: 0.9906 - val_precision: 0.9908 - val_recall: 0.9903
Epoch 9/100
439/439 [==============================] - 8s 19ms/step - loss: 0.0137 - 
accuracy: 0.9956 - precision: 0.9960 - recall: 0.9954 - val_loss: 0.0362 
- val_accuracy: 0.9899 - val_precision: 0.9906 - val_recall: 0.9899
Epoch 10/100
439/439 [==============================] - 8s 19ms/step - loss: 0.0112 - 
accuracy: 0.9961 - precision: 0.9966 - recall: 0.9959 - val_loss: 0.0365 
- val_accuracy: 0.9903 - val_precision: 0.9909 - val_recall: 0.9902
Epoch 00010: early stopping
Test loss: 0.04324812442064285
Test accuracy: 0.9902856945991516
Test precision: 0.9904244542121887
Test recall: 0.9900000095367432
Test f1 score: 0.9902857142857143
Test AUC for digit: 0 0.9962574850299402
Test AUC for digit: 1 0.9940907921236538
Test AUC for digit: 2 0.9984028163786178
Test AUC for digit: 3 0.993960028357141
Test AUC for digit: 4 0.9914325054702
Test AUC for digit: 5 0.9978038687780799
Test AUC for digit: 6 0.9957936507936508
Test AUC for digit: 7 0.9913140793939336
Test AUC for digit: 8 0.9926190476190476
Test AUC for digit: 9 0.9948151881227197
```





Next we increased number of filters. This caused a raise of number of parameters to over 700 thousands but model did not perform better than basic model. Test accuracy was 99%, which is slightly less than basic model's accuracy. It means that we should not use such high number of filters because we do not need them. This model also seems to be overfitted.

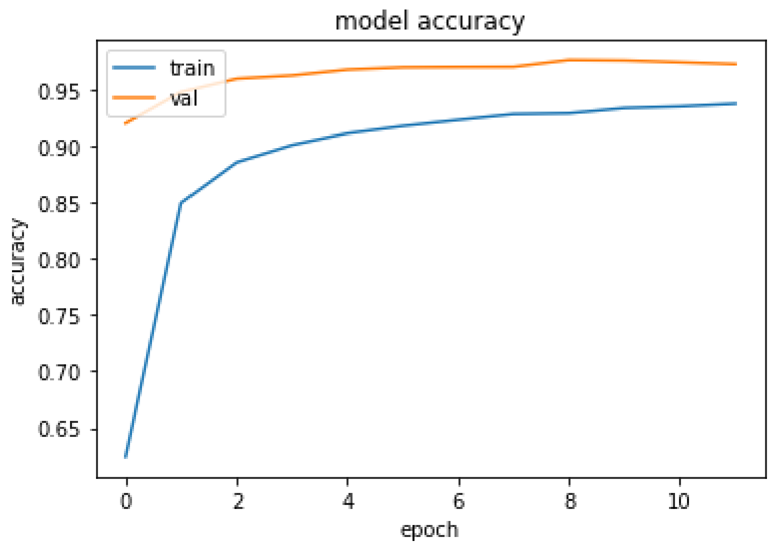
Different size and type of pooling layers

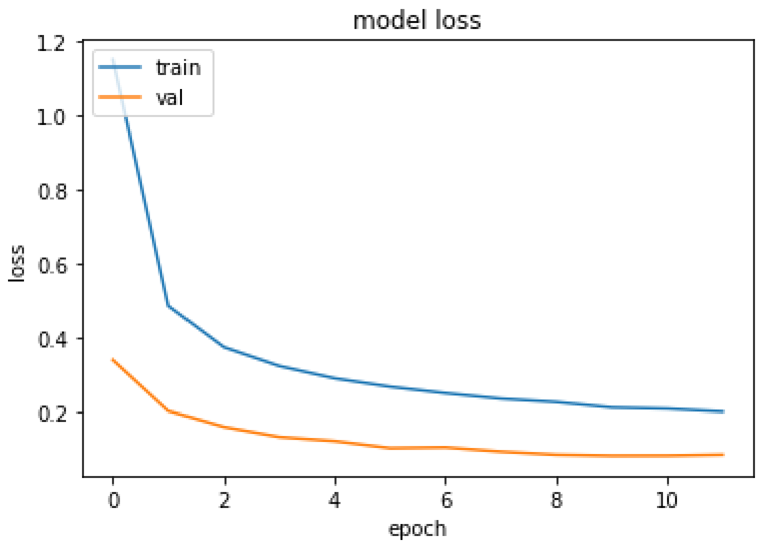
In [41]:
```
model_cnn_pool5 = keras.Sequential([
    layers.Conv2D(32, (3,3), activation="relu", input_shape=(28,28,1)),
    layers.MaxPooling2D (5,3),
    layers.Conv2D(64, (3,3), activation="relu"),
    layers.MaxPooling2D (5,3),

    layers.Flatten(),
    layers.Dropout(.5),
    layers.Dense(10, activation="softmax")
])
model_cnn_pool5.summary()
predict_model(model_cnn_pool5, [es], epochs=100)
```

```
Model: "sequential_17"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_13 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 6, 6, 64)          18496
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 1, 1, 64)          0
_________________________________________________________________
flatten_17 (Flatten)         (None, 64)                0
_________________________________________________________________
dropout_16 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_54 (Dense)             (None, 10)                650
=================================================================
Total params: 19,466
Trainable params: 19,466
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
439/439 [==============================] - 17s 25ms/step - loss: 1.6386 -
accuracy: 0.4443 - precision: 0.7621 - recall: 0.1895 - val_loss: 0.3402
- val_accuracy: 0.9205 - val_precision: 0.9659 - val_recall: 0.8592
Epoch 2/100
439/439 [==============================] - 3s 8ms/step - loss: 0.5372 - a
ccuracy: 0.8335 - precision: 0.8979 - recall: 0.7583 - val_loss: 0.2031 -
val_accuracy: 0.9482 - val_precision: 0.9721 - val_recall: 0.9242
Epoch 3/100
439/439 [==============================] - 3s 8ms/step - loss: 0.3899 - a
ccuracy: 0.8815 - precision: 0.9224 - recall: 0.8415 - val_loss: 0.1599 -
val_accuracy: 0.9600 - val_precision: 0.9747 - val_recall: 0.9439
Epoch 4/100
439/439 [==============================] - 3s 7ms/step - loss: 0.3313 - a
ccuracy: 0.8988 - precision: 0.9294 - recall: 0.8673 - val_loss: 0.1329 -
val_accuracy: 0.9629 - val_precision: 0.9762 - val_recall: 0.9489
Epoch 5/100
439/439 [==============================] - 3s 7ms/step - loss: 0.2954 - a
ccuracy: 0.9092 - precision: 0.9359 - recall: 0.8834 - val_loss: 0.1220 -
val_accuracy: 0.9681 - val_precision: 0.9795 - val_recall: 0.9576
Epoch 6/100
439/439 [==============================] - 3s 8ms/step - loss: 0.2813 - a
ccuracy: 0.9143 - precision: 0.9392 - recall: 0.8897 - val_loss: 0.1032 -
val_accuracy: 0.9701 - val_precision: 0.9799 - val_recall: 0.9633
Epoch 7/100
439/439 [==============================] - 3s 8ms/step - loss: 0.2496 - a
ccuracy: 0.9239 - precision: 0.9440 - recall: 0.9045 - val_loss: 0.1051 -
val_accuracy: 0.9704 - val_precision: 0.9797 - val_recall: 0.9593
Epoch 8/100
439/439 [==============================] - 3s 7ms/step - loss: 0.2431 - a
ccuracy: 0.9248 - precision: 0.9447 - recall: 0.9058 - val_loss: 0.0936 -
val_accuracy: 0.9706 - val_precision: 0.9798 - val_recall: 0.9641
Epoch 9/100
439/439 [==============================] - 3s 7ms/step - loss: 0.2384 - a
ccuracy: 0.9260 - precision: 0.9459 - recall: 0.9090 - val_loss: 0.0853 -
val_accuracy: 0.9766 - val_precision: 0.9830 - val_recall: 0.9696
Epoch 10/100
439/439 [==============================] - 3s 7ms/step - loss: 0.2130 - a
ccuracy: 0.9333 - precision: 0.9492 - recall: 0.9185 - val_loss: 0.0826 -
val_accuracy: 0.9762 - val_precision: 0.9835 - val_recall: 0.9711
Epoch 11/100
439/439 [==============================] - 3s 7ms/step - loss: 0.2164 - a
ccuracy: 0.9316 - precision: 0.9492 - recall: 0.9175 - val_loss: 0.0828 -
val_accuracy: 0.9746 - val_precision: 0.9807 - val_recall: 0.9701
Epoch 12/100
439/439 [==============================] - 3s 7ms/step - loss: 0.2055 - a
ccuracy: 0.9370 - precision: 0.9515 - recall: 0.9241 - val_loss: 0.0853 -
val_accuracy: 0.9730 - val_precision: 0.9796 - val_recall: 0.9698
```

```
Epoch 00012: early stopping
Test loss: 0.09769929945468903
Test accuracy: 0.9715714454650879
Test precision: 0.9770860075950623
Test recall: 0.9685714244842529
Test f1 score: 0.9715714285714285
Test AUC for digit: 0 0.9948361331663899
Test AUC for digit: 1 0.9941565446942954
Test AUC for digit: 2 0.9841899519327072
Test AUC for digit: 3 0.9719724776720643
Test AUC for digit: 4 0.9860265364292893
Test AUC for digit: 5 0.9800909396406366
Test AUC for digit: 6 0.9928571428571429
Test AUC for digit: 7 0.9823115528040143
Test AUC for digit: 8 0.9796031746031746
Test AUC for digit: 9 0.9761439939197929
```





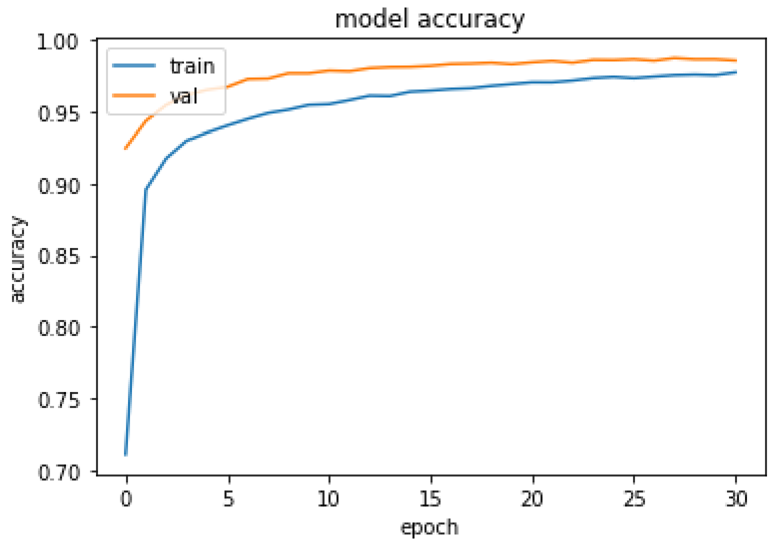
Next, we checked different size of pooling layers. We decided to create a MaxPooling layers with size equals (5,5) and stride equals 3. It means that we take a square of values with size 5x5 then we look for max value, we write it in the middle of square and than we "move" 3 numbers in right or down direction. As we can see, the accuracy is worse than basic model, because we lose too much information in MaxPooling layers. The plots also shows that this model is underfitted.

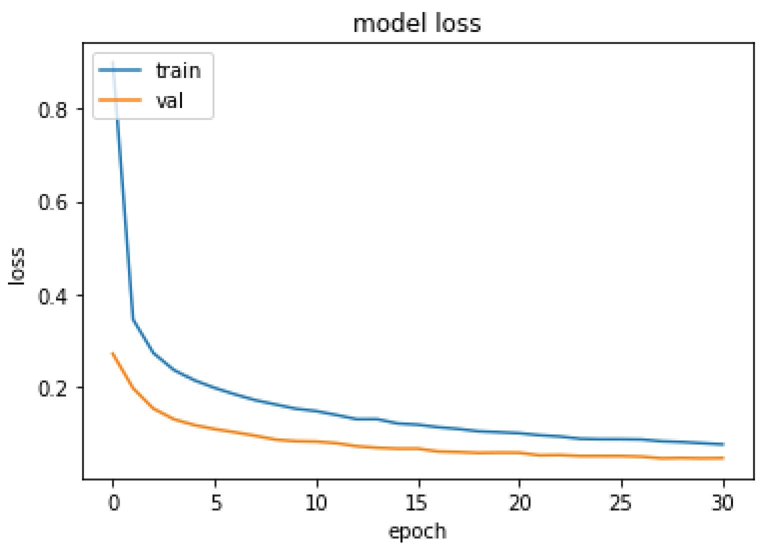
In [42]:
```
model_cnn_avg = keras.Sequential([
    layers.Conv2D(32, (3,3), activation="relu", input_shape=(28,28,1)),
    layers.AveragePooling2D (3,3),
    layers.Conv2D(64, (3,3), activation="relu"),
    layers.AveragePooling2D (3,3),

    layers.Flatten(),
    layers.Dropout(.5),
    layers.Dense(10, activation="softmax")
])
model_cnn_avg.summary()
predict_model(model_cnn_avg, [es], epochs=100)
```

```
Model: "sequential_18"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_15 (Conv2D)           (None, 26, 26, 32)        320       
_________________________________________________________________
average_pooling2d_2 (Average (None, 8, 8, 32)          0         
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 6, 6, 64)          18496     
_________________________________________________________________
average_pooling2d_3 (Average (None, 2, 2, 64)          0         
_________________________________________________________________
flatten_18 (Flatten)         (None, 256)               0         
_________________________________________________________________
dropout_17 (Dropout)         (None, 256)               0         
_________________________________________________________________
dense_55 (Dense)             (None, 10)                2570      
=================================================================
Total params: 21,386
Trainable params: 21,386
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
439/439 [==============================] - 4s 8ms/step - loss: 1.4358 - a
ccuracy: 0.5197 - precision: 0.7783 - recall: 0.3063 - val_loss: 0.2705 -
val_accuracy: 0.9248 - val_precision: 0.9531 - val_recall: 0.8922
Epoch 2/100
439/439 [==============================] - 3s 8ms/step - loss: 0.3783 - a
ccuracy: 0.8837 - precision: 0.9196 - recall: 0.8476 - val_loss: 0.1960 -
val_accuracy: 0.9437 - val_precision: 0.9590 - val_recall: 0.9251
Epoch 3/100
439/439 [==============================] - 3s 7ms/step - loss: 0.2821 - a
ccuracy: 0.9156 - precision: 0.9367 - recall: 0.8952 - val_loss: 0.1518 -
val_accuracy: 0.9548 - val_precision: 0.9683 - val_recall: 0.9423
Epoch 4/100
439/439 [==============================] - 3s 7ms/step - loss: 0.2438 - a
ccuracy: 0.9272 - precision: 0.9439 - recall: 0.9095 - val_loss: 0.1290 -
val_accuracy: 0.9618 - val_precision: 0.9702 - val_recall: 0.9535
Epoch 5/100
439/439 [==============================] - 3s 7ms/step - loss: 0.2236 - a
ccuracy: 0.9325 - precision: 0.9481 - recall: 0.9185 - val_loss: 0.1161 -
val_accuracy: 0.9654 - val_precision: 0.9729 - val_recall: 0.9576
Epoch 6/100
439/439 [==============================] - 3s 7ms/step - loss: 0.2054 - a
ccuracy: 0.9364 - precision: 0.9496 - recall: 0.9246 - val_loss: 0.1074 -
val_accuracy: 0.9674 - val_precision: 0.9733 - val_recall: 0.9622
Epoch 7/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1861 - a
ccuracy: 0.9435 - precision: 0.9551 - recall: 0.9341 - val_loss: 0.1005 -
val_accuracy: 0.9729 - val_precision: 0.9769 - val_recall: 0.9655
Epoch 8/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1737 - a
ccuracy: 0.9474 - precision: 0.9569 - recall: 0.9373 - val_loss: 0.0930 -
val_accuracy: 0.9732 - val_precision: 0.9797 - val_recall: 0.9664
Epoch 9/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1644 - a
ccuracy: 0.9495 - precision: 0.9593 - recall: 0.9412 - val_loss: 0.0846 -
```

```
val_accuracy: 0.9769 - val_precision: 0.9813 - val_recall: 0.9713
Epoch 10/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1554 - a
ccuracy: 0.9541 - precision: 0.9624 - recall: 0.9459 - val_loss: 0.0813 -
val_accuracy: 0.9769 - val_precision: 0.9796 - val_recall: 0.9720
Epoch 11/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1508 - a
ccuracy: 0.9546 - precision: 0.9632 - recall: 0.9462 - val_loss: 0.0804 -
val_accuracy: 0.9788 - val_precision: 0.9818 - val_recall: 0.9733
Epoch 12/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1422 - a
ccuracy: 0.9573 - precision: 0.9656 - recall: 0.9500 - val_loss: 0.0769 -
val_accuracy: 0.9784 - val_precision: 0.9820 - val_recall: 0.9747
Epoch 13/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1312 - a
ccuracy: 0.9601 - precision: 0.9681 - recall: 0.9534 - val_loss: 0.0703 -
val_accuracy: 0.9805 - val_precision: 0.9837 - val_recall: 0.9772
Epoch 14/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1299 - a
ccuracy: 0.9603 - precision: 0.9674 - recall: 0.9546 - val_loss: 0.0670 -
val_accuracy: 0.9812 - val_precision: 0.9850 - val_recall: 0.9778
Epoch 15/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1196 - a
ccuracy: 0.9636 - precision: 0.9695 - recall: 0.9576 - val_loss: 0.0651 -
val_accuracy: 0.9814 - val_precision: 0.9845 - val_recall: 0.9785
Epoch 16/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1184 - a
ccuracy: 0.9636 - precision: 0.9692 - recall: 0.9584 - val_loss: 0.0651 -
val_accuracy: 0.9821 - val_precision: 0.9855 - val_recall: 0.9779
Epoch 17/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1094 - a
ccuracy: 0.9654 - precision: 0.9711 - recall: 0.9599 - val_loss: 0.0589 -
val_accuracy: 0.9834 - val_precision: 0.9855 - val_recall: 0.9808
Epoch 18/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1074 - a
ccuracy: 0.9668 - precision: 0.9724 - recall: 0.9627 - val_loss: 0.0578 -
val_accuracy: 0.9837 - val_precision: 0.9866 - val_recall: 0.9811
Epoch 19/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1010 - a
ccuracy: 0.9683 - precision: 0.9735 - recall: 0.9638 - val_loss: 0.0559 -
val_accuracy: 0.9843 - val_precision: 0.9865 - val_recall: 0.9817
Epoch 20/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0975 - a
ccuracy: 0.9704 - precision: 0.9751 - recall: 0.9660 - val_loss: 0.0564 -
val_accuracy: 0.9833 - val_precision: 0.9867 - val_recall: 0.9817
Epoch 21/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0961 - a
ccuracy: 0.9713 - precision: 0.9754 - recall: 0.9670 - val_loss: 0.0562 -
val_accuracy: 0.9846 - val_precision: 0.9865 - val_recall: 0.9821
Epoch 22/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0930 - a
ccuracy: 0.9708 - precision: 0.9752 - recall: 0.9669 - val_loss: 0.0506 -
val_accuracy: 0.9856 - val_precision: 0.9887 - val_recall: 0.9843
Epoch 23/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0942 - a
ccuracy: 0.9710 - precision: 0.9755 - recall: 0.9667 - val_loss: 0.0512 -
val_accuracy: 0.9843 - val_precision: 0.9869 - val_recall: 0.9821
Epoch 24/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0892 - a
ccuracy: 0.9731 - precision: 0.9770 - recall: 0.9693 - val_loss: 0.0492 -
val_accuracy: 0.9863 - val_precision: 0.9884 - val_recall: 0.9846
Epoch 25/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0863 - a
ccuracy: 0.9739 - precision: 0.9774 - recall: 0.9702 - val_loss: 0.0492 -
val_accuracy: 0.9861 - val_precision: 0.9881 - val_recall: 0.9844
Epoch 26/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0864 - a
ccuracy: 0.9734 - precision: 0.9771 - recall: 0.9697 - val_loss: 0.0491 -
```

```
val_accuracy: 0.9867 - val_precision: 0.9886 - val_recall: 0.9850
Epoch 27/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0884 - a
ccuracy: 0.9740 - precision: 0.9777 - recall: 0.9707 - val_loss: 0.0478 -
val_accuracy: 0.9857 - val_precision: 0.9877 - val_recall: 0.9835
Epoch 28/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0799 - a
ccuracy: 0.9758 - precision: 0.9791 - recall: 0.9725 - val_loss: 0.0442 -
val_accuracy: 0.9876 - val_precision: 0.9894 - val_recall: 0.9863
Epoch 29/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0820 - a
ccuracy: 0.9755 - precision: 0.9789 - recall: 0.9724 - val_loss: 0.0449 -
val_accuracy: 0.9867 - val_precision: 0.9891 - val_recall: 0.9856
Epoch 30/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0779 - a
ccuracy: 0.9748 - precision: 0.9784 - recall: 0.9718 - val_loss: 0.0441 -
val_accuracy: 0.9867 - val_precision: 0.9887 - val_recall: 0.9848
Epoch 31/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0730 - a
ccuracy: 0.9779 - precision: 0.9811 - recall: 0.9751 - val_loss: 0.0446 -
val_accuracy: 0.9859 - val_precision: 0.9877 - val_recall: 0.9844
Epoch 00031: early stopping
Test loss: 0.05266290158033371
Test accuracy: 0.9857142567634583
Test precision: 0.9866762161254883
Test recall: 0.9838571548461914
Test f1 score: 0.9857142857142858
Test AUC for digit: 0 0.9975965630331252
Test AUC for digit: 1 0.9964337348028355
Test AUC for digit: 2 0.9899322237602509
Test AUC for digit: 3 0.9902563737312167
Test AUC for digit: 4 0.993108146858049
Test AUC for digit: 5 0.992317975361152
Test AUC for digit: 6 0.9943650793650793
Test AUC for digit: 7 0.9950732539562421
Test AUC for digit: 8 0.9821428571428572
Test AUC for digit: 9 0.9892405775544455
```





After, we changed MaxPooling layer to AveragePooling layer. The difference between this two layers is that AveragePooling layer sums the values in the square and divides by the number of values in square. Results are worse than basic model because MaxPooling, by its characteristics, is better when we have black background, because it remembers the most white value in grey-scale.

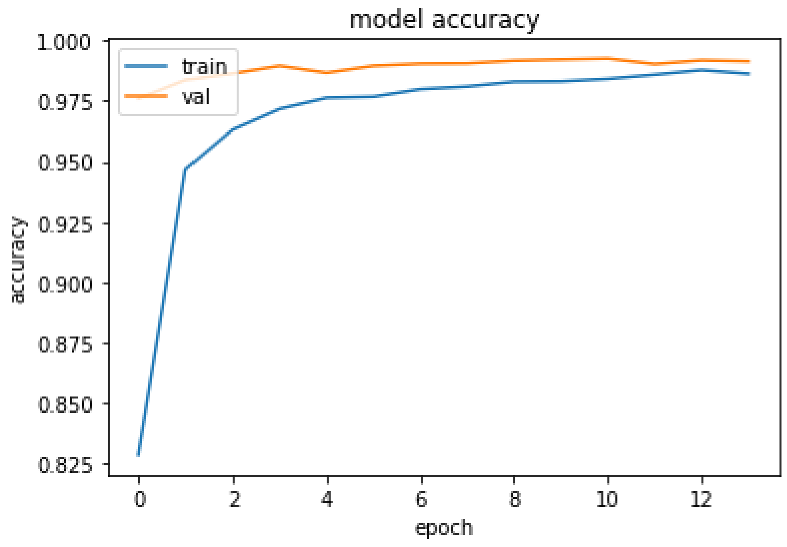
Different number of full conected layers

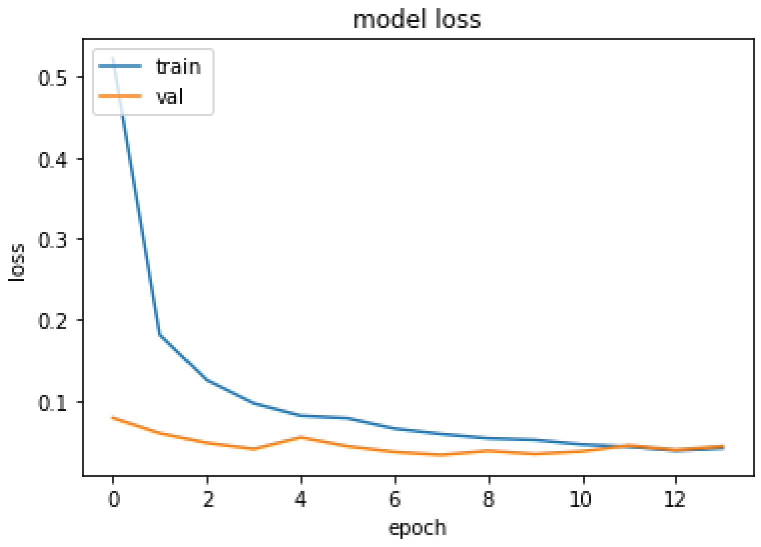
In [44]:

```
model_cnn_fc = keras.Sequential([
    layers.Conv2D(32, (3,3), activation="relu", input_shape=(28,28,1)),
    layers.MaxPooling2D (2,2),
    layers.Conv2D(64, (3,3), activation="relu"),
    layers.MaxPooling2D (2,2),

    layers.Flatten(),
    layers.Dense(128, activation="relu"),
    layers.Dense(32, activation="relu"),
    layers.Dropout(.5),
    layers.Dense(10, activation="softmax")
])
model_cnn_fc.summary()
predict_model(model_cnn_fc, [es], epochs=100)
```

```
Model: "sequential_19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_17 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 5, 5, 64)          0
_________________________________________________________________
flatten_19 (Flatten)         (None, 1600)              0
_________________________________________________________________
dense_56 (Dense)             (None, 128)               204928
_________________________________________________________________
dense_57 (Dense)             (None, 32)                4128
_________________________________________________________________
dropout_18 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_58 (Dense)             (None, 10)                330
```

```
=================================================================
Total params: 228,202
Trainable params: 228,202
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
439/439 [==============================] - 5s 9ms/step - loss: 0.9434 - a
ccuracy: 0.6736 - precision: 0.8816 - recall: 0.5434 - val_loss: 0.0790 -
val_accuracy: 0.9759 - val_precision: 0.9819 - val_recall: 0.9717
Epoch 2/100
439/439 [==============================] - 4s 8ms/step - loss: 0.2156 - a
ccuracy: 0.9378 - precision: 0.9588 - recall: 0.9172 - val_loss: 0.0600 -
val_accuracy: 0.9835 - val_precision: 0.9864 - val_recall: 0.9815
Epoch 3/100
439/439 [==============================] - 3s 8ms/step - loss: 0.1334 - a
ccuracy: 0.9598 - precision: 0.9733 - recall: 0.9482 - val_loss: 0.0479 -
val_accuracy: 0.9863 - val_precision: 0.9893 - val_recall: 0.9843
Epoch 4/100
439/439 [==============================] - 3s 8ms/step - loss: 0.0965 - a
ccuracy: 0.9712 - precision: 0.9809 - recall: 0.9635 - val_loss: 0.0405 -
val_accuracy: 0.9895 - val_precision: 0.9916 - val_recall: 0.9879
Epoch 5/100
439/439 [==============================] - 3s 8ms/step - loss: 0.0784 - a
ccuracy: 0.9771 - precision: 0.9837 - recall: 0.9699 - val_loss: 0.0548 -
val_accuracy: 0.9866 - val_precision: 0.9877 - val_recall: 0.9853
Epoch 6/100
439/439 [==============================] - 3s 8ms/step - loss: 0.0962 - a
ccuracy: 0.9722 - precision: 0.9800 - recall: 0.9644 - val_loss: 0.0438 -
val_accuracy: 0.9895 - val_precision: 0.9899 - val_recall: 0.9889
Epoch 7/100
439/439 [==============================] - 3s 8ms/step - loss: 0.0635 - a
ccuracy: 0.9800 - precision: 0.9857 - recall: 0.9746 - val_loss: 0.0368 -
val_accuracy: 0.9903 - val_precision: 0.9909 - val_recall: 0.9895
Epoch 8/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0566 - a
ccuracy: 0.9808 - precision: 0.9863 - recall: 0.9768 - val_loss: 0.0331 -
val_accuracy: 0.9905 - val_precision: 0.9918 - val_recall: 0.9893
Epoch 9/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0508 - a
ccuracy: 0.9838 - precision: 0.9878 - recall: 0.9795 - val_loss: 0.0382 -
val_accuracy: 0.9916 - val_precision: 0.9922 - val_recall: 0.9909
Epoch 10/100
439/439 [==============================] - 3s 8ms/step - loss: 0.0570 - a
ccuracy: 0.9811 - precision: 0.9860 - recall: 0.9768 - val_loss: 0.0343 -
val_accuracy: 0.9921 - val_precision: 0.9931 - val_recall: 0.9915
Epoch 11/100
439/439 [==============================] - 3s 8ms/step - loss: 0.0483 - a
ccuracy: 0.9835 - precision: 0.9882 - recall: 0.9795 - val_loss: 0.0376 -
val_accuracy: 0.9925 - val_precision: 0.9932 - val_recall: 0.9921
Epoch 12/100
439/439 [==============================] - 3s 8ms/step - loss: 0.0452 - a
ccuracy: 0.9859 - precision: 0.9890 - recall: 0.9819 - val_loss: 0.0450 -
val_accuracy: 0.9902 - val_precision: 0.9910 - val_recall: 0.9896
Epoch 13/100
439/439 [==============================] - 3s 8ms/step - loss: 0.0371 - a
ccuracy: 0.9883 - precision: 0.9905 - recall: 0.9848 - val_loss: 0.0395 -
val_accuracy: 0.9918 - val_precision: 0.9922 - val_recall: 0.9909
Epoch 14/100
439/439 [==============================] - 3s 8ms/step - loss: 0.0443 - a
ccuracy: 0.9852 - precision: 0.9887 - recall: 0.9817 - val_loss: 0.0439 -
val_accuracy: 0.9913 - val_precision: 0.9918 - val_recall: 0.9908
Epoch 00014: early stopping
Test loss: 0.06664026528596878
Test accuracy: 0.9888571500778198
Test precision: 0.9892780780792236
Test recall: 0.9885714054107666
Test f1 score: 0.9888571428571429
Test AUC for digit: 0 0.9990146050287297
Test AUC for digit: 1 0.9951085363883655
Test AUC for digit: 2 0.9946951426069947
Test AUC for digit: 3 0.9941206967376037
Test AUC for digit: 4 0.9936140097241216
Test AUC for digit: 5 0.9941211483938233
Test AUC for digit: 6 0.9940476190476191
Test AUC for digit: 7 0.9912349278979704
Test AUC for digit: 8 0.9904761904761904
Test AUC for digit: 9 0.9916870128535711
```





The performance of a published network (LeNet5, VGG, Yolo, etc) for recognizing MNIST Digits

We decided to implement the architecture of LeNet5 network. The LeNet-5 architecture consists of two sets of convolutional and average pooling layers, followed by a flattening convolutional layer, then two fully-connected layers and finally a softmax classifier.

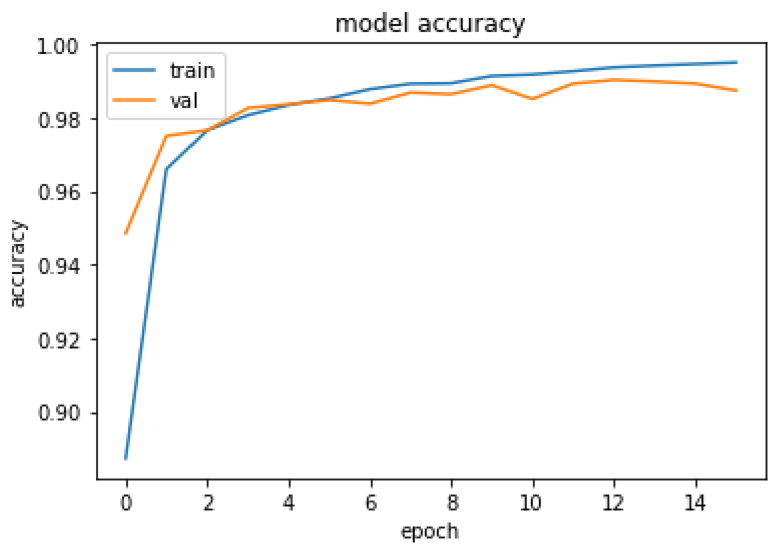
LeNet5

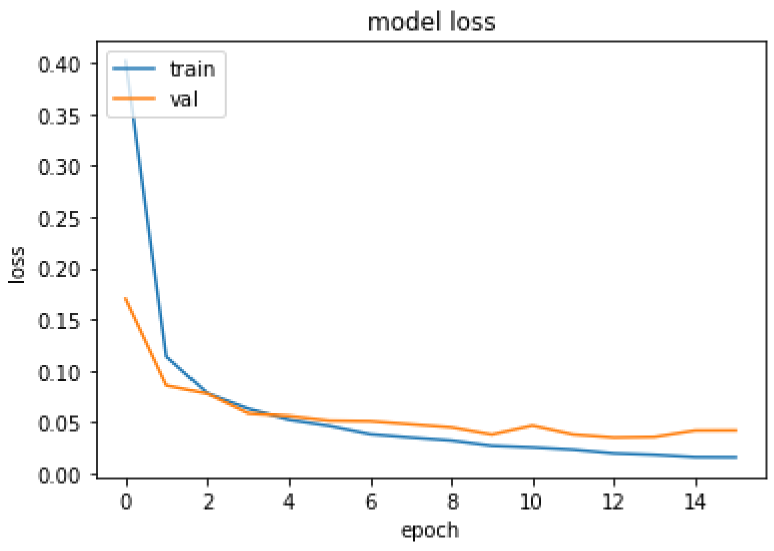
This is an implementation of LeNet5 (slightly different, because input shape is 28x28 and in original version was 32x32). Despite of its age the model is pretty accurate (with accuracy 98.9%). This is close to our best models, and it does not have too big number of parameters (only 60,074).

In [66]:
```
lenet5 = keras.Sequential([
    layers.Conv2D(filters=6, kernel_size=(3, 3), activation='relu', inpu
    layers.AveragePooling2D(),

    layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'),
    layers.AveragePooling2D(),

    layers.Flatten(),

    layers.Dense(units=120, activation='relu'),
    layers.Dense(units=84, activation='relu'),
    layers.Dense(units=10, activation='softmax')
])

lenet5.summary()
predict_model(lenet5, [es], epochs=100)
```

```
Model: "sequential_27"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_30 (Conv2D)           (None, 26, 26, 6)         60
_________________________________________________________________
average_pooling2d_12 (Averag (None, 13, 13, 6)         0
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 11, 11, 16)        880
_________________________________________________________________
average_pooling2d_13 (Averag (None, 5, 5, 16)          0
_________________________________________________________________
flatten_27 (Flatten)         (None, 400)               0
_________________________________________________________________
dense_82 (Dense)             (None, 120)               48120
_________________________________________________________________
dense_83 (Dense)             (None, 84)                10164
_________________________________________________________________
dense_84 (Dense)             (None, 10)                850
=================================================================
Total params: 60,074
Trainable params: 60,074
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
439/439 [==============================] - 4s 7ms/step - loss: 0.8082 - a
ccuracy: 0.7743 - precision: 0.8916 - recall: 0.6138 - val_loss: 0.1701 -
val_accuracy: 0.9485 - val_precision: 0.9569 - val_recall: 0.9416
Epoch 2/100
439/439 [==============================] - 3s 7ms/step - loss: 0.1352 - a
ccuracy: 0.9603 - precision: 0.9669 - recall: 0.9535 - val_loss: 0.0855 -
val_accuracy: 0.9750 - val_precision: 0.9798 - val_recall: 0.9723
Epoch 3/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0792 - a
ccuracy: 0.9762 - precision: 0.9797 - recall: 0.9734 - val_loss: 0.0775 -
val_accuracy: 0.9766 - val_precision: 0.9808 - val_recall: 0.9743
Epoch 4/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0660 - a
ccuracy: 0.9803 - precision: 0.9830 - recall: 0.9780 - val_loss: 0.0585 -
val_accuracy: 0.9827 - val_precision: 0.9855 - val_recall: 0.9812
Epoch 5/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0530 - a
ccuracy: 0.9835 - precision: 0.9857 - recall: 0.9815 - val_loss: 0.0560 -
val_accuracy: 0.9837 - val_precision: 0.9865 - val_recall: 0.9815
Epoch 6/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0451 - a
ccuracy: 0.9858 - precision: 0.9874 - recall: 0.9846 - val_loss: 0.0515 -
val_accuracy: 0.9848 - val_precision: 0.9877 - val_recall: 0.9838
Epoch 7/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0382 - a
ccuracy: 0.9878 - precision: 0.9892 - recall: 0.9867 - val_loss: 0.0510 -
```

```
val_accuracy: 0.9838 - val_precision: 0.9872 - val_recall: 0.9825
Epoch 8/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0330 - a
ccuracy: 0.9898 - precision: 0.9910 - recall: 0.9890 - val_loss: 0.0480 -
val_accuracy: 0.9869 - val_precision: 0.9881 - val_recall: 0.9848
Epoch 9/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0357 - a
ccuracy: 0.9880 - precision: 0.9894 - recall: 0.9870 - val_loss: 0.0450 -
val_accuracy: 0.9864 - val_precision: 0.9880 - val_recall: 0.9848
Epoch 10/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0265 - a
ccuracy: 0.9913 - precision: 0.9920 - recall: 0.9902 - val_loss: 0.0379 -
val_accuracy: 0.9889 - val_precision: 0.9899 - val_recall: 0.9879
Epoch 11/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0245 - a
ccuracy: 0.9916 - precision: 0.9922 - recall: 0.9911 - val_loss: 0.0468 -
val_accuracy: 0.9851 - val_precision: 0.9865 - val_recall: 0.9843
Epoch 12/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0253 - a
ccuracy: 0.9920 - precision: 0.9927 - recall: 0.9915 - val_loss: 0.0380 -
val_accuracy: 0.9893 - val_precision: 0.9905 - val_recall: 0.9883
Epoch 13/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0206 - a
ccuracy: 0.9935 - precision: 0.9939 - recall: 0.9928 - val_loss: 0.0350 -
val_accuracy: 0.9903 - val_precision: 0.9913 - val_recall: 0.9887
Epoch 14/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0168 - a
ccuracy: 0.9947 - precision: 0.9953 - recall: 0.9943 - val_loss: 0.0356 -
val_accuracy: 0.9899 - val_precision: 0.9909 - val_recall: 0.9893
Epoch 15/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0133 - a
ccuracy: 0.9955 - precision: 0.9959 - recall: 0.9952 - val_loss: 0.0419 -
val_accuracy: 0.9893 - val_precision: 0.9900 - val_recall: 0.9883
Epoch 16/100
439/439 [==============================] - 3s 7ms/step - loss: 0.0129 - a
ccuracy: 0.9959 - precision: 0.9962 - recall: 0.9957 - val_loss: 0.0420 -
val_accuracy: 0.9874 - val_precision: 0.9884 - val_recall: 0.9870
Restoring model weights from the end of the best epoch.
Epoch 00016: early stopping
Test loss: 0.039734747260808945
Test accuracy: 0.9887142777442932
Test precision: 0.9896981120109558
Test recall: 0.9881428480148315
Test f1 score: 0.9887142857142858
Test AUC for digit: 0 0.9967844920645857
Test AUC for digit: 1 0.9953059136050564
Test AUC for digit: 2 0.9949662450022941
Test AUC for digit: 3 0.9916327889937758
Test AUC for digit: 4 0.9944156815297652
Test AUC for digit: 5 0.9953975922796036
Test AUC for digit: 6 0.9949879467017794
Test AUC for digit: 7 0.9939484045292735
Test AUC for digit: 8 0.986462230229477
Test AUC for digit: 9 0.9932852101356561
```





The best network

In [117…

```
from sklearn.metrics import confusion_matrix

y_pred = model_cnn_long.predict(x_test)
y_pred1 = list(np.argmax(y_pred, axis=1))
y_test1 = list(np.argmax(y_test, axis = 1))

confusion_matrix = confusion_matrix(y_test1, y_pred1)

print(confusion_matrix)
```

```
[[654   0   0   0   0   0   0   0   0   0]
 [  0 765   0   0   0   0   0   1   0   1]
 [  1   1 697   1   0   0   0   3   1   0]
 [  0   0   0 707   0   0   0   2   0   0]
 [  0   0   0   0 670   0   1   0   0   2]
 [  1   1   0   0   0 648   0   0   0   2]
 [  1   0   0   1   0   1 697   0   0   0]
 [  0   0   0   0   0   1   0 745   0   0]
 [  0   0   0   1   0   0   0   0 683   2]
 [  0   1   0   0   3   1   0   1   0 703]]
```

We chose model with best accuracy. It was a model with 4 convolutional layers and this is confusion matrix of this model.

In [ ]: