```
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import numpy as np
import os as os
import math
import matplotlib
import matplotlib.pyplot as plt
import sklearn
import sklearn.metrics as sm
import seaborn as sn

print(os.getcwd())
```

```
    /content
```

```
print(os.getcwd())
path_file = '/content'
path_data = '/content/data_nndl'
# set path to data and load data
## not pd.read_csv('datafile.csv') because we need array not data frame
os.chdir(path_data)
os.getcwd()


x_train = np.genfromtxt(path_data + '/csvTrainImages 60k x 784.csv', delimiter = ',')
y_train = np.genfromtxt(path_data + '/csvTrainLabel 60k x 1.csv', delimiter = ',')
x_test = np.genfromtxt(path_data + '/csvTestImages 10k x 784.csv', delimiter = ',')
y_test = np.genfromtxt(path_data + '/csvTestLabel 10k x 1.csv', delimiter = ',')

# convert to float (pixel values are integers)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')

# convert class vectors to binary class matrices
num_classes = 10 # 10 digits to classify, 0 - 9
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# reset path to original directory
os.chdir(path_file)
os.getcwd()
```

```
    /content
    '/content'
```

Data Description

- arabic numbers, handwritten by 700 participants, each of them writing the number 0-9 for a total of 10 times.
- each in a image file of yz x yx pixels
- this results in 70k observations, for each observation we have a image of said pixel size and a labe of what the participant acutally wanted to write.
- data already partitioned into train and test data sets and converted into CSV files for easier accesaability.
- CSV contains the flattend array of pixel values.

```
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

```
((60000, 784), (60000, 10), (10000, 784), (10000, 10))
```

Dimensions of the training and test data set look good. We have 60k image observations for training and validation, and keep 10k observations for the final test of the neural net.

To make sure there where no fautly conversion, we plot a couple of numbers from the CSV flattend array and compare them to the acutal images.

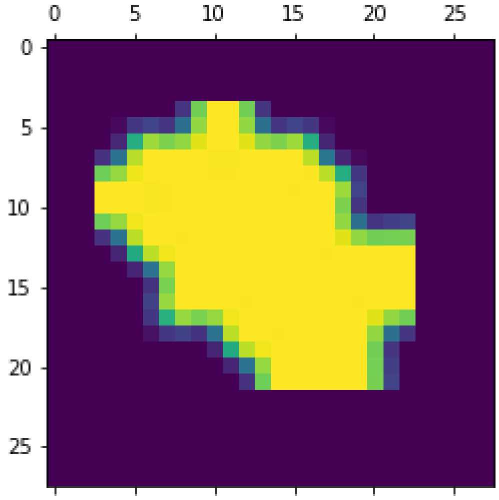
```
x_train_2d = np.reshape(x_train,(60000,28,28))
x_train_2d.shape
# shows that now we have no longer a o-dimensional array of 60k x 784 but the restorec
```

```
(60000, 28, 28)
```

```
plt.matshow(x_train_2d[0])

# PLUS ACTUAL IMAGE
```

```
<matplotlib.image.AxesImage at 0x7f968b1ca450>
```



```
y_train[4]
```

```
    array([0., 0., 0., 0., 1., 0., 0., 0., 0., 0.], dtype=float32)
```

having hade a discussion about data proceed now to formulate the first neural network.

sources for different parameters

- rob's lectures
- youtube tutorial => https://www.youtube.com/watch?v=iqQgED9vV7k&ab_channel=codebasics
- batch size differences => https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e

Zum Bearbeiten doppelklicken (oder Eingabe)

1st neural net

To start we begin with a simple 1 layer NN. We choose a dense layer, using 10 nodes, giving space to classify each of the available digits from 0-9. to keep it simple we use sigmoid ad a activation function.

```
model1 = keras.Sequential([
    keras.layers.Dense(10, input_shape=(x_train.shape[1],), activation='sigmoid'),
])
model1.summary()
model1.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
    Model: "sequential_38"
    _________________________________________________________________
    Layer (type)                 Output Shape              Param #   
    =================================================================
    dense_99 (Dense)             (None, 10)                7850      
    =================================================================
    Total params: 7,850
    Trainable params: 7,850
    Non-trainable params: 0
    _________________________________________________________________
```

```
model1.fit(x_train, y_train,
          batch_size = 128,
          epochs = 5,
              validation_split = 0.3,
              shuffle = True,
              verbose = 1)

#validation_data=(x_test, y_test)) ??

print('> model evaluation')
model1.evaluate(x_test, y_test, verbose = 1)
```

```
Epoch 1/5
329/329 [==============================] - 1s 2ms/step - loss: 8.2457 - accuracy
Epoch 2/5
329/329 [==============================] - 1s 2ms/step - loss: 2.3136 - accuracy
Epoch 3/5
329/329 [==============================] - 1s 2ms/step - loss: 1.6467 - accuracy
Epoch 4/5
329/329 [==============================] - 1s 2ms/step - loss: 1.4836 - accuracy
Epoch 5/5
329/329 [==============================] - 1s 2ms/step - loss: 1.3377 - accuracy
> model evaluation
313/313 [==============================] - 0s 703us/step - loss: 2.4515 - accura
[2.451476573944092, 0.9401999711990356]
```

2nd neural net

mentioned in the lecture slides and also youtube sources, standardizing the pixel values for a range of 0-1 should improve accuracy

```
x_train_stan = x_train / 255
x_test_stan = x_test / 255


model2 = keras.Sequential([
    keras.layers.Dense(10, input_shape=(x_train_stan.shape[1],), activation='sigmoid'
])
model2.summary()
model2.compile(optimizer='adam',
              loss= 'categorical_crossentropy',
              metrics=['accuracy'])
```

```
Model: "sequential_39"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_100 (Dense)            (None, 10)                7850      
=================================================================
Total params: 7,850
Trainable params: 7,850
Non-trainable params: 0
_________________________________________________________________
```

```
model2.fit(x_train_stan, y_train,
           batch_size = 128,
           epochs = 5,
           #validation_data=(x_test, y_test)),
           validation_split = 0.3,
           shuffle = True,
           verbose = 1)

print('> model evaluation')
model2.evaluate(x_test_stan, y_test, verbose = 1)
```

```
Epoch 1/5
329/329 [==============================] - 1s 2ms/step - loss: 0.6027 - accuracy
Epoch 2/5
329/329 [==============================] - 1s 2ms/step - loss: 0.2254 - accuracy
Epoch 3/5
329/329 [==============================] - 1s 2ms/step - loss: 0.1740 - accuracy
Epoch 4/5
329/329 [==============================] - 1s 2ms/step - loss: 0.1493 - accuracy
Epoch 5/5
329/329 [==============================] - 1s 2ms/step - loss: 0.1342 - accuracy
> model evaluation
313/313 [==============================] - 0s 712us/step - loss: 0.1739 - accura
[0.1738690882921219, 0.9538000226020813]
```

no decrease in accuracy so we continue henceforth with the standardized pixel values.

but accuracy is still improvable, thus we want to experiment with a multi-layerd neural net.

3rd neural net

following the exercise jupyter notebook from the lecture 3 (mnist_mlp.ipynb) we try a model with more layers

- relu for activatzion function, should be better.
- soft max better for classification, dense10 at the end remains => makes sense

```
model3 = keras.Sequential([
    keras.layers.Dense(512, input_shape=(x_train_stan.shape[1],), activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(512, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(10, activation='softmax'),
])
model3.summary() ;
model3.compile(optimizer='adam',
               loss='categorical_crossentropy',
               metrics=['accuracy'])
```

```
              metrics=['accuracy'])
```

```
Model: "sequential_40"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_101 (Dense)            (None, 512)               401920    
_________________________________________________________________
dropout_58 (Dropout)         (None, 512)               0         
_________________________________________________________________
dense_102 (Dense)            (None, 512)               262656    
_________________________________________________________________
dropout_59 (Dropout)         (None, 512)               0         
_________________________________________________________________
dense_103 (Dense)            (None, 10)                5130      
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
```

```
model3.fit(x_train_stan, y_train,
           batch_size = 128,
           epochs = 5,
           #validation_data=(x_test, y_test)),
           validation_split = 0.3,
           shuffle = True,
           verbose = 1)

print('> model evaluation')
model3.evaluate(x_test_stan, y_test, verbose = 1)
```

```
Epoch 1/5
329/329 [==============================] - 5s 13ms/step - loss: 0.1769 - accuracy
Epoch 2/5
329/329 [==============================] - 4s 13ms/step - loss: 0.0566 - accuracy
Epoch 3/5
329/329 [==============================] - 4s 13ms/step - loss: 0.0400 - accuracy
Epoch 4/5
329/329 [==============================] - 4s 12ms/step - loss: 0.0261 - accuracy
Epoch 5/5
329/329 [==============================] - 4s 13ms/step - loss: 0.0225 - accuracy
> model evaluation
313/313 [==============================] - 1s 2ms/step - loss: 0.0709 - accuracy
[0.07091709226369858, 0.9807999730110168]
```

4th neural net dense only

Data still 1D array, additional Flatten() layers

```
model4 = keras.Sequential([
    keras.layers.Dense(512, input_shape=(x_train_stan.shape[1],), activation='relu'),
    keras.layers.Dense(512, input_shape=(x_train_stan.shape[1],), activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Flatten(),
    keras.layers.Dense(512, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Flatten(),
    keras.layers.Dense(10, activation='softmax'),
])

model4.compile(optimizer='adam',
              loss= 'categorical_crossentropy',
              metrics=['accuracy'])

model4.fit(x_train_stan, y_train,
           batch_size = 128,
           epochs = 10,
           validation_split = 0.3,
           shuffle = True,
           verbose = 1)

print('> model evaluation')
model4.evaluate(x_test_stan, y_test, verbose = 1)
```

```
Epoch 1/10
329/329 [==============================] - 5s 13ms/step - loss: 0.1701 - accurac
Epoch 2/10
329/329 [==============================] - 4s 13ms/step - loss: 0.0563 - accurac
Epoch 3/10
329/329 [==============================] - 4s 13ms/step - loss: 0.0374 - accurac
Epoch 4/10
329/329 [==============================] - 4s 13ms/step - loss: 0.0264 - accurac
Epoch 5/10
329/329 [==============================] - 4s 12ms/step - loss: 0.0229 - accurac
Epoch 6/10
329/329 [==============================] - 4s 13ms/step - loss: 0.0211 - accurac
Epoch 7/10
329/329 [==============================] - 4s 12ms/step - loss: 0.0136 - accurac
Epoch 8/10
329/329 [==============================] - 4s 12ms/step - loss: 0.0146 - accurac
Epoch 9/10
329/329 [==============================] - 4s 13ms/step - loss: 0.0138 - accurac
Epoch 10/10
329/329 [==============================] - 4s 13ms/step - loss: 0.0121 - accurac
> model evaluation
313/313 [==============================] - 1s 2ms/step - loss: 0.0830 - accuracy
[0.08301497995853424, 0.9822999835014343]
```

5th neural net

still 1d array with flatten layers and many more dense layers, declining in size

```
model5 = keras.Sequential([
    keras.layers.Dense(512, input_shape=(x_train_stan.shape[1],), activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Flatten(),
    keras.layers.Dense(342, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Flatten(),
    keras.layers.Dense(225, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Flatten(),
    keras.layers.Dense(135, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Flatten(),
    keras.layers.Dense(81, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Flatten(),    keras.layers.Dense(10, activation='softmax'),
])

model5.compile(optimizer='adam',
              loss= 'categorical_crossentropy',
              metrics=['accuracy'])

model5.fit(x_train_stan, y_train,
          batch_size = 128,
          epochs = 10,
          #validation_data=(x_test, y_test)),
          validation_split = 0.3,
          shuffle = True,
          verbose = 1)

print('> model evaluation')
model5.evaluate(x_test_stan, y_test, verbose = 1)
```

```
Epoch 1/10
329/329 [==============================] - 15s 45ms/step - loss: 0.2727 - accura
Epoch 2/10
329/329 [==============================] - 5s 14ms/step - loss: 0.0803 - accurac
Epoch 3/10
329/329 [==============================] - 5s 14ms/step - loss: 0.0590 - accurac
Epoch 4/10
329/329 [==============================] - 5s 14ms/step - loss: 0.0449 - accurac
Epoch 5/10
329/329 [==============================] - 5s 14ms/step - loss: 0.0376 - accurac
Epoch 6/10
329/329 [==============================] - 5s 14ms/step - loss: 0.0399 - accurac
Epoch 7/10
329/329 [==============================] - 5s 14ms/step - loss: 0.0270 - accurac
Epoch 8/10
329/329 [==============================] - 4s 14ms/step - loss: 0.0258 - accurac
Epoch 9/10
329/329 [==============================] - 4s 14ms/step - loss: 0.0255 - accurac
```

```
Epoch 10/10
329/329 [==============================] - 4s 14ms/step - loss: 0.0190 - accurac
> model evaluation
313/313 [==============================] - 1s 2ms/step - loss: 0.1002 - accuracy
[0.10022571682929993, 0.9818000197410583]
```

CONVOLUTIONAL NEURAL NETS

following lecture slides, we learned that 2d arrrays work better for number recognition as they can be read by a convolutional network

following the lecture slides, we know that convoluted networks are better suited for image classification. following the exercise jupyter notebook from the (mnist_cnn.ipynb) we try a model with more layers

relu for activatzion function, should be better. soft max better for classification, dense10 at the end remains => makes sens

for convolution nets to work we need to restructre the data back to multidimensional array from the flattened version we imported from *csv*

```
x_train_2d_stan = np.reshape(x_train_stan,(60000,28,28,1))
x_test_2d_stan = np.reshape(x_test_stan,(10000,28,28,1))
```

Conv 1st neural net

```
num_filters = 12
filter_size = 5
pool_size = 2

modelc1 = keras.Sequential([
  keras.layers.Conv2D(num_filters, filter_size, input_shape=(28, 28, 1)),
  keras.layers.MaxPooling2D(pool_size=pool_size),
  keras.layers.Flatten(),
  keras.layers.Dense(10, activation='softmax'),
])

modelc1.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

modelc1.fit(x_train_2d_stan, y_train,
          validation_split = 0.3,
          shuffle = True,
          epochs= 15,
          verbose = 1)

modelc1.evaluate(x_test_2d_stan, y_test, verbose = 1)

     Epoch 1/15
     1313/1313 [==============================] - 15s 11ms/step - loss: 0.1580 - accu
     Epoch 2/15
     1313/1313 [==============================] - 14s 11ms/step - loss: 0.0694 - accu
     Epoch 3/15
     1313/1313 [==============================] - 14s 11ms/step - loss: 0.0490 - accu
     Epoch 4/15
     1313/1313 [==============================] - 14s 11ms/step - loss: 0.0374 - accu
     Epoch 5/15
     1313/1313 [==============================] - 14s 11ms/step - loss: 0.0303 - accu
     Epoch 6/15
     1313/1313 [==============================] - 14s 11ms/step - loss: 0.0241 - accu
     Epoch 7/15
     1313/1313 [==============================] - 14s 11ms/step - loss: 0.0206 - accu
     Epoch 8/15
     1313/1313 [==============================] - 14s 11ms/step - loss: 0.0177 - accu
     Epoch 9/15
     1313/1313 [==============================] - 14s 11ms/step - loss: 0.0149 - accu
     Epoch 10/15
     1313/1313 [==============================] - 14s 11ms/step - loss: 0.0134 - accu
     Epoch 11/15
     1313/1313 [==============================] - 14s 11ms/step - loss: 0.0113 - accu
     Epoch 12/15
     1313/1313 [==============================] - 14s 11ms/step - loss: 0.0108 - accu
     Epoch 13/15
     1313/1313 [==============================] - 14s 11ms/step - loss: 0.0098 - accu
     Epoch 14/15
     1313/1313 [==============================] - 14s 11ms/step - loss: 0.0089 - accu
     Epoch 15/15
     1313/1313 [==============================] - 14s 11ms/step - loss: 0.0081 - accu
     313/313 [==============================] - 1s 4ms/step - loss: 0.0524 - accuracy
     [0.05243486911058426, 0.9869999885559082]
```

Conv 2nd neural net

2d array, bigger kernel used

```
#  2D,  grössser kernel Conv2D eingebaut
modelc2 = keras.Sequential([
    keras.layers.Conv2D(32, kernel_size=(6, 6),
                activation='relu',
                input_shape=(28,28,1)),
    keras.layers.MaxPooling2D(pool_size=(5, 5)),
    keras.layers.Flatten(),
    keras.layers.Dense(10, activation='softmax'),
])

modelc2.compile(optimizer='adam',
modelc2.compile(optimizer='adam',
              loss= 'categorical_crossentropy',
              metrics=['accuracy'])

modelc2.fit(x_train_2d_stan, y_train,
           batch_size = 128,
           epochs = 10,
           #validation_data=(x_test, y_test)),
           validation_split = 0.3,
           shuffle = True,
           verbose = 1)

print('> model evaluation')
modelc2.evaluate(x_test_2d_stan, y_test, verbose = 1)
```

```
Epoch 1/10
329/329 [==============================] - 14s 42ms/step - loss: 0.3892 - accura
Epoch 2/10
329/329 [==============================] - 14s 41ms/step - loss: 0.0623 - accura
Epoch 3/10
329/329 [==============================] - 13s 41ms/step - loss: 0.0457 - accura
Epoch 4/10
329/329 [==============================] - 13s 41ms/step - loss: 0.0371 - accura
Epoch 5/10
329/329 [==============================] - 14s 41ms/step - loss: 0.0327 - accura
Epoch 6/10
329/329 [==============================] - 13s 41ms/step - loss: 0.0287 - accura
Epoch 7/10
329/329 [==============================] - 13s 41ms/step - loss: 0.0266 - accura
Epoch 8/10
329/329 [==============================] - 13s 41ms/step - loss: 0.0244 - accura
Epoch 9/10
329/329 [==============================] - 13s 41ms/step - loss: 0.0230 - accura
Epoch 10/10
329/329 [==============================] - 13s 41ms/step - loss: 0.0210 - accura
> model evaluation
313/313 [==============================] - 1s 5ms/step - loss: 0.0395 - accuracy
[0.03949955478310585, 0.9873999953269958]
```

Conv 3rd neural net

bigger kernel, more dense layers at the end

```
#  2D,  grössseres Conv2D eingebaut, mit mehr dense am ende
modelc3 = keras.Sequential([
    keras.layers.Conv2D(32, kernel_size=(6, 6),
                 activation='relu',
                 input_shape=(28,28,1)),
    keras.layers.MaxPooling2D(pool_size=(5, 5)),
    keras.layers.Dropout(0.25),
    keras.layers.Flatten(),
    keras.layers.Dense(255, activation='relu'),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax'),
])

modelc3.compile(optimizer='adam',
              loss= 'categorical_crossentropy',
              metrics=['accuracy'])

modelc3.fit(x_train_2d_stan, y_train,
           batch_size = 128,
           epochs = 10,
           #validation_data=(x_test, y_test)),
           validation_split = 0.3,
           shuffle = True,
           verbose = 1)

print('> model evaluation')
modelc3.evaluate(x_test_2d_stan, y_test, verbose = 1)
```

```
Epoch 1/10
329/329 [==============================] - 15s 45ms/step - loss: 0.1954 - accura
Epoch 2/10
329/329 [==============================] - 15s 46ms/step - loss: 0.0452 - accura
Epoch 3/10
329/329 [==============================] - 15s 46ms/step - loss: 0.0363 - accura
Epoch 4/10
329/329 [==============================] - 15s 46ms/step - loss: 0.0287 - accura
Epoch 5/10
329/329 [==============================] - 15s 46ms/step - loss: 0.0255 - accura
Epoch 6/10
329/329 [==============================] - 15s 45ms/step - loss: 0.0217 - accura
Epoch 7/10
329/329 [==============================] - 15s 46ms/step - loss: 0.0209 - accura
Epoch 8/10
329/329 [==============================] - 15s 46ms/step - loss: 0.0191 - accura
Epoch 9/10
329/329 [==============================] - 15s 45ms/step - loss: 0.0164 - accura
Epoch 10/10
329/329 [==============================] - 15s 45ms/step - loss: 0.0166 - accura
> model evaluation
313/313 [==============================] - 2s 5ms/step - loss: 0.0285 - accuracy
[0.028519269078969955, 0.9914000034332275]
```

Conv 4th neural net

bigger kernel, 3 kernels, then dense, dropout and flatten layers, padding of 1 pixel

```
#  2D,  grösssseres und mehr Conv2D eingebaut
modelc4 = keras.Sequential([
    keras.layers.ZeroPadding2D(padding=(1,1)),
    keras.layers.ZeroPadding2D(padding=(1,1)),
    keras.layers.Conv2D(32, kernel_size=(14, 14),
                activation='relu',
                input_shape=(28,28,1)),
    keras.layers.Conv2D(64, kernel_size=(6, 6),activation='relu'),
    keras.layers.Conv2D(128, kernel_size=(3, 3),activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Dropout(0.2),
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(10, activation='softmax'),
])

modelc4.compile(optimizer='adam',
              loss= 'categorical_crossentropy',
              metrics=['accuracy'])

modelc4.fit(x_train_2d_stan, y_train,
          batch_size = 128,
          epochs = 15,
          #validation_data=(x_test, y_test)),
          validation_split = 0.3,
          shuffle = True,
          verbose = 1)

print('> model evaluation')
modelc4.evaluate(x_test_2d_stan, y_test, verbose = 1)
```

```
Epoch 1/15
329/329 [==============================] - 117s 352ms/step - loss: 0.2474 - accu
Epoch 2/15
329/329 [==============================] - 115s 351ms/step - loss: 0.0607 - accu
Epoch 3/15
329/329 [==============================] - 115s 350ms/step - loss: 0.0461 - accu
Epoch 4/15
329/329 [==============================] - 115s 349ms/step - loss: 0.0432 - accu
Epoch 5/15
329/329 [==============================] - 115s 349ms/step - loss: 0.0349 - accu
Epoch 6/15
329/329 [==============================] - 115s 350ms/step - loss: 0.0314 - accu
Epoch 7/15
329/329 [==============================] - 115s 350ms/step - loss: 0.0293 - accu
Epoch 8/15
329/329 [==============================] - 115s 349ms/step - loss: 0.0279 - accu
Epoch 9/15
329/329 [==============================] - 115s 349ms/step - loss: 0.0219 - accu
Epoch 10/15
329/329 [==============================] - 115s 349ms/step - loss: 0.0234 - accu
Epoch 11/15
329/329 [==============================] - 115s 350ms/step - loss: 0.0200 - accu
Epoch 12/15
```

```
329/329 [==============================] - 115s 348ms/step - loss: 0.0175 - accu
Epoch 13/15
329/329 [==============================] - 114s 346ms/step - loss: 0.0190 - accu
Epoch 14/15
329/329 [==============================] - 113s 344ms/step - loss: 0.0193 - accu
Epoch 15/15
329/329 [==============================] - 113s 345ms/step - loss: 0.0129 - accu
> model evaluation
313/313 [==============================] - 7s 23ms/step - loss: 0.0476 - accurac
[0.047644421458244324, 0.9890000224113464]
```

Conv 5th neural net

padding of 2 pixels, smaller kernels but much more of them

```
#  2D,  kleiner aber mehrere layers Conv2D eingebaut
modelc5 = keras.Sequential([
    keras.layers.ZeroPadding2D(padding=(2,2)),
    keras.layers.Conv2D(18, kernel_size=(6, 6),
                activation='relu',
                input_shape=(28,28,1)),
    keras.layers.Conv2D(32, kernel_size=(5, 5),activation='relu'),
    keras.layers.Conv2D(18, kernel_size=(4, 4),activation='relu'),
    keras.layers.Conv2D(32, kernel_size=(3, 3),activation='relu'),
    keras.layers.Conv2D(32, kernel_size=(2, 2),activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Dropout(0.2),
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(10, activation='softmax'),
])

modelc5.compile(optimizer='adam',
              loss= 'categorical_crossentropy',
              metrics=['accuracy'])

modelc5.fit(x_train_2d_stan, y_train,
           batch_size = 128,
           epochs = 15,
           #validation_data=(x_test, y_test)),
           validation_split = 0.3,
           shuffle = True,
           verbose = 1)

print('> model evaluation')
modelc5.evaluate(x_test_2d_stan, y_test, verbose = 1)
```

```
Epoch 1/15
```

```
329/329 [==============================] - 134s 406ms/step - loss: 0.1823 - accu
Epoch 2/15
329/329 [==============================] - 133s 405ms/step - loss: 0.0409 - accu
Epoch 3/15
329/329 [==============================] - 133s 405ms/step - loss: 0.0322 - accu
Epoch 4/15
329/329 [==============================] - 133s 405ms/step - loss: 0.0274 - accu
Epoch 5/15
329/329 [==============================] - 133s 406ms/step - loss: 0.0213 - accu
Epoch 6/15
329/329 [==============================] - 133s 404ms/step - loss: 0.0180 - accu
Epoch 7/15
329/329 [==============================] - 133s 404ms/step - loss: 0.0197 - accu
Epoch 8/15
329/329 [==============================] - 133s 404ms/step - loss: 0.0161 - accu
Epoch 9/15
329/329 [==============================] - 133s 404ms/step - loss: 0.0151 - accu
Epoch 10/15
329/329 [==============================] - 133s 404ms/step - loss: 0.0145 - accu
Epoch 11/15
329/329 [==============================] - 133s 405ms/step - loss: 0.0121 - accu
Epoch 12/15
329/329 [==============================] - 133s 405ms/step - loss: 0.0128 - accu
Epoch 13/15
329/329 [==============================] - 133s 405ms/step - loss: 0.0118 - accu
Epoch 14/15
329/329 [==============================] - 133s 405ms/step - loss: 0.0117 - accu
Epoch 15/15
329/329 [==============================] - 133s 405ms/step - loss: 0.0108 - accu
> model evaluation
313/313 [==============================] - 8s 26ms/step - loss: 0.0463 - accurac
[0.04625444486737251, 0.9883000254631042]
```

Conv 6th neural net

kernel size of conv 2nd net, more dense layers at the end.

```
#  2D,  grösssseres Conv2D eingebaut, mit noch mehr dense am ende
modelc6 = keras.Sequential([
    keras.layers.Conv2D(32, kernel_size=(6, 6),
                activation='relu',
                input_shape=(28,28,1)),
    keras.layers.MaxPooling2D(pool_size=(5, 5)),
    keras.layers.Dropout(0.2),
    keras.layers.Flatten(),
    keras.layers.Dense(255, activation='relu'),
    keras.layers.Dropout(0.1),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout(0.1),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dropout(0.1),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dropout(0.1),

    keras.layers.Dense(10, activation='softmax'),
])

modelc6.compile(optimizer='adam',
              loss= 'categorical_crossentropy',
              metrics=['accuracy'])

modelc6.fit(x_train_2d_stan, y_train,
           batch_size = 128,
           epochs = 15,
           #validation_data=(x_test, y_test)),
           validation_split = 0.3,
           shuffle = True,
           verbose = 1)

print('> model evaluation')
modelc6.evaluate(x_test_2d_stan, y_test, verbose = 1)
```

```
Epoch 1/15
329/329 [==============================] - 16s 47ms/step - loss: 0.3587 - accura
Epoch 2/15
329/329 [==============================] - 15s 46ms/step - loss: 0.0630 - accura
Epoch 3/15
329/329 [==============================] - 15s 46ms/step - loss: 0.0446 - accura
Epoch 4/15
329/329 [==============================] - 15s 46ms/step - loss: 0.0378 - accura
Epoch 5/15
329/329 [==============================] - 15s 46ms/step - loss: 0.0334 - accura
Epoch 6/15
329/329 [==============================] - 15s 46ms/step - loss: 0.0272 - accura
Epoch 7/15
329/329 [==============================] - 15s 46ms/step - loss: 0.0280 - accura
Epoch 8/15
329/329 [==============================] - 15s 46ms/step - loss: 0.0248 - accura
Epoch 9/15
329/329 [==============================] - 15s 46ms/step - loss: 0.0226 - accura
Epoch 10/15
329/329 [==============================] - 15s 46ms/step - loss: 0.0201 - accura
Epoch 11/15
329/329 [==============================] - 15s 46ms/step - loss: 0.0180 - accura
Epoch 12/15
329/329 [==============================] - 15s 46ms/step - loss: 0.0191 - accura
Epoch 13/15
329/329 [==============================] - 15s 46ms/step - loss: 0.0184 - accura
Epoch 14/15
329/329 [==============================] - 15s 46ms/step - loss: 0.0176 - accura
Epoch 15/15
329/329 [==============================] - 15s 46ms/step - loss: 0.0154 - accura
> model evaluation
313/313 [==============================] - 2s 5ms/step - loss: 0.0399 - accuracy
[0.039946287870407104, 0.9902999997138977]
```

PARAMETERS SPECIFICATION

having found 3rd model to have the highest accuracy in testing, we continue to evaluate different parameters for the model compilation.

different loss function

loss = keras.losses.categorical_crossentropy

```
model = keras.Sequential([
    keras.layers.Conv2D(32, kernel_size=(6, 6),
                activation='relu',
                input_shape=(28,28,1)),
    keras.layers.MaxPooling2D(pool_size=(5, 5)),
    keras.layers.Dropout(0.25),
    keras.layers.Flatten(),
    keras.layers.Dense(255, activation='relu'),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax'),
])

model.compile(optimizer='adam',
              loss=keras.losses.categorical_crossentropy,
              metrics=['accuracy'])

model.fit(x_train_2d_stan, y_train,
           batch_size = 128,
           epochs = 15,
           #validation_data=(x_test, y_test)),
           validation_split = 0.3,
           shuffle = True,
           verbose = 1)

print('> model evaluation')
model.evaluate(x_test_2d_stan, y_test, verbose = 1)
```

```
    Epoch 1/15
    329/329 [==============================] - 15s 46ms/step - loss: 0.2037 - accura
    Epoch 2/15
    329/329 [==============================] - 15s 45ms/step - loss: 0.0469 - accura
    Epoch 3/15
    329/329 [==============================] - 15s 45ms/step - loss: 0.0335 - accura
    Epoch 4/15
    329/329 [==============================] - 15s 45ms/step - loss: 0.0279 - accura
    Epoch 5/15
    329/329 [==============================] - 15s 45ms/step - loss: 0.0239 - accura
    Epoch 6/15
    329/329 [==============================] - 15s 45ms/step - loss: 0.0215 - accura
```

```
Epoch 7/15
329/329 [==============================] - 15s 45ms/step - loss: 0.0207 - accura
Epoch 8/15
329/329 [==============================] - 15s 45ms/step - loss: 0.0200 - accura
Epoch 9/15
329/329 [==============================] - 15s 45ms/step - loss: 0.0174 - accura
Epoch 10/15
329/329 [==============================] - 15s 45ms/step - loss: 0.0148 - accura
Epoch 11/15
329/329 [==============================] - 15s 45ms/step - loss: 0.0133 - accura
Epoch 12/15
329/329 [==============================] - 15s 45ms/step - loss: 0.0150 - accura
Epoch 13/15
329/329 [==============================] - 15s 44ms/step - loss: 0.0130 - accura
Epoch 14/15
329/329 [==============================] - 15s 45ms/step - loss: 0.0127 - accura
Epoch 15/15
329/329 [==============================] - 15s 45ms/step - loss: 0.0117 - accura
> model evaluation
313/313 [==============================] - 2s 6ms/step - loss: 0.0308 - accuracy
[0.030769916251301765, 0.9914000034332275]
```

different loss function, different optimizer

optimizer=keras.optimizers.Adadelta(), loss=keras.losses.categorical_crossentropy

```
# 2D,  grösssseres Conv2D eingebaut, mit mehr dense am ende
# andere loss function, andere optimizer function

model = keras.Sequential([
    keras.layers.Conv2D(32, kernel_size=(6, 6),
                 activation='relu',
                 input_shape=(28,28,1)),
    keras.layers.MaxPooling2D(pool_size=(5, 5)),
    keras.layers.Dropout(0.25),
    keras.layers.Flatten(),
    keras.layers.Dense(255, activation='relu'),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax'),
])

model.compile(optimizer=keras.optimizers.Adadelta(),
              loss=keras.losses.categorical_crossentropy,
              metrics=['accuracy'])

model.fit(x_train_2d_stan, y_train,
          batch_size = 128,
          epochs = 15,
          #validation_data=(x_test, y_test)),
          validation_split = 0.3,
           shuffle = True,
           verbose = 1)

print('> model evaluation')
model.evaluate(x_test_2d_stan, y_test, verbose = 1)
```

```
Epoch 1/15
329/329 [==============================] - 15s 46ms/step - loss: 2.3070 - accura
Epoch 2/15
329/329 [==============================] - 15s 45ms/step - loss: 2.2892 - accura
Epoch 3/15
329/329 [==============================] - 15s 45ms/step - loss: 2.2711 - accura
Epoch 4/15
329/329 [==============================] - 15s 45ms/step - loss: 2.2523 - accura
Epoch 5/15
329/329 [==============================] - 15s 45ms/step - loss: 2.2339 - accura
Epoch 6/15
329/329 [==============================] - 15s 45ms/step - loss: 2.2137 - accura
Epoch 7/15
329/329 [==============================] - 15s 45ms/step - loss: 2.1918 - accura
Epoch 8/15
329/329 [==============================] - 15s 45ms/step - loss: 2.1692 - accura
Epoch 9/15
329/329 [==============================] - 15s 45ms/step - loss: 2.1444 - accura
Epoch 10/15
329/329 [==============================] - 15s 45ms/step - loss: 2.1175 - accura
Epoch 11/15
329/329 [==============================] - 15s 45ms/step - loss: 2.0878 - accura
Epoch 12/15
329/329 [==============================] - 15s 45ms/step - loss: 2.0563 - accura
Epoch 13/15
329/329 [==============================] - 15s 45ms/step - loss: 2.0223 - accura
Epoch 14/15
329/329 [==============================] - 15s 45ms/step - loss: 1.9863 - accura
Epoch 15/15
329/329 [==============================] - 15s 45ms/step - loss: 1.9475 - accura
> model evaluation
313/313 [==============================] - 2s 5ms/step - loss: 1.8971 - accuracy
[1.8971047401428223, 0.7743999958038330]
```

different loss function, differnent batch_size

loss = keras.losses.categorical_crossentropy batch_size = 512

```
model = keras.Sequential([
    keras.layers.Conv2D(32, kernel_size=(6, 6),
                 activation='relu',
                 input_shape=(28,28,1)),
    keras.layers.MaxPooling2D(pool_size=(5, 5)),
    keras.layers.Dropout(0.25),
    keras.layers.Flatten(),
    keras.layers.Dense(255, activation='relu'),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax'),
])

model.compile(optimizer='adam',
              loss=keras.losses.categorical_crossentropy,
              metrics=['accuracy'])

model.fit(x_train_2d_stan, y_train,
           batch_size = 512,
           epochs = 15,
           #validation_data=(x_test, y_test)),
           validation_split = 0.3,
           shuffle = True,
           verbose = 1)

print('> model evaluation')
model.evaluate(x_test_2d_stan, y_test, verbose = 1)
```

```
Epoch 1/15
83/83 [==============================] - 15s 170ms/step - loss: 0.5036 - accurac
Epoch 2/15
83/83 [==============================] - 14s 163ms/step - loss: 0.0831 - accurac
Epoch 3/15
83/83 [==============================] - 13s 163ms/step - loss: 0.0571 - accurac
Epoch 4/15
83/83 [==============================] - 13s 163ms/step - loss: 0.0447 - accurac
Epoch 5/15
83/83 [==============================] - 13s 163ms/step - loss: 0.0427 - accurac
Epoch 6/15
83/83 [==============================] - 13s 162ms/step - loss: 0.0328 - accurac
Epoch 7/15
83/83 [==============================] - 13s 162ms/step - loss: 0.0306 - accurac
Epoch 8/15
83/83 [==============================] - 13s 162ms/step - loss: 0.0286 - accurac
Epoch 9/15
83/83 [==============================] - 13s 161ms/step - loss: 0.0267 - accurac
Epoch 10/15
83/83 [==============================] - 13s 162ms/step - loss: 0.0250 - accurac
Epoch 11/15
83/83 [==============================] - 13s 161ms/step - loss: 0.0229 - accurac
Epoch 12/15
83/83 [==============================] - 13s 162ms/step - loss: 0.0193 - accurac
Epoch 13/15
83/83 [==============================] - 13s 161ms/step - loss: 0.0196 - accurac
Epoch 14/15
83/83 [==============================] - 13s 161ms/step - loss: 0.0193 - accurac
Epoch 15/15
83/83 [==============================] - 13s 161ms/step - loss: 0.0187 - accurac
> model evaluation
313/313 [==============================] - 2s 5ms/step - loss: 0.0274 - accuracy
[0.027373697608709335, 0.9909999966621399]
```

only different batch_size

batch_size = 512

```
# 2D,  grösssseres Conv2D eingebaut, mit mehr dense am ende
# nur andere batch size

model = keras.Sequential([
    keras.layers.Conv2D(32, kernel_size=(6, 6),
                 activation='relu',
                 input_shape=(28,28,1)),
    keras.layers.MaxPooling2D(pool_size=(5, 5)),
    keras.layers.Dropout(0.25),
    keras.layers.Flatten(),
    keras.layers.Dense(255, activation='relu'),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax'),
])

model.compile(optimizer='adam',
              loss= 'categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train_2d_stan, y_train,
           batch_size = 512,
           epochs = 15,
           #validation_data=(x_test, y_test)),
           validation_split = 0.3,
           shuffle = True,
           verbose = 1)

print('> model evaluation')
model.evaluate(x_test_2d_stan, y_test, verbose = 1)
```

```
Epoch 1/15
83/83 [==============================] - 14s 164ms/step - loss: 0.4474 - accurac
Epoch 2/15
83/83 [==============================] - 13s 162ms/step - loss: 0.0705 - accurac
Epoch 3/15
83/83 [==============================] - 13s 162ms/step - loss: 0.0497 - accurac
Epoch 4/15
83/83 [==============================] - 13s 162ms/step - loss: 0.0454 - accurac
Epoch 5/15
83/83 [==============================] - 13s 162ms/step - loss: 0.0343 - accurac
Epoch 6/15
83/83 [==============================] - 13s 161ms/step - loss: 0.0342 - accurac
Epoch 7/15
83/83 [==============================] - 13s 161ms/step - loss: 0.0303 - accurac
Epoch 8/15
83/83 [==============================] - 13s 161ms/step - loss: 0.0255 - accurac
```

```
Epoch 9/15
83/83 [==============================] - 13s 161ms/step - loss: 0.0245 - accurac
Epoch 10/15
83/83 [==============================] - 13s 162ms/step - loss: 0.0208 - accurac
Epoch 11/15
83/83 [==============================] - 13s 161ms/step - loss: 0.0198 - accurac
Epoch 12/15
83/83 [==============================] - 13s 161ms/step - loss: 0.0215 - accurac
Epoch 13/15
83/83 [==============================] - 13s 162ms/step - loss: 0.0265 - accurac
Epoch 14/15
83/83 [==============================] - 13s 161ms/step - loss: 0.0175 - accurac
Epoch 15/15
83/83 [==============================] - 13s 161ms/step - loss: 0.0159 - accurac
> model evaluation
313/313 [==============================] - 2s 5ms/step - loss: 0.0319 - accuracy
[0.03186453878879547, 0.9908000230789185]
```

only different batch_size

2000

```
model = keras.Sequential([
    keras.layers.Conv2D(32, kernel_size=(6, 6),
                 activation='relu',
                 input_shape=(28,28,1)),
    keras.layers.MaxPooling2D(pool_size=(5, 5)),
    keras.layers.Dropout(0.25),
    keras.layers.Flatten(),
    keras.layers.Dense(255, activation='relu'),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax'),
])

model.compile(optimizer='adam',
              loss= 'categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train_2d_stan, y_train,
           batch_size = 2000,
           epochs = 15,
           #validation_data=(x_test, y_test)),
           validation_split = 0.3,
           shuffle = True,
           verbose = 1)

print('> model evaluation')
model.evaluate(x_test_2d_stan, y_test, verbose = 1)
```

```
Epoch 1/15
21/21 [==============================] - 13s 605ms/step - loss: 1.3476 - accurac
Epoch 2/15
21/21 [==============================] - 13s 599ms/step - loss: 0.1945 - accurac
Epoch 3/15
21/21 [==============================] - 13s 599ms/step - loss: 0.1084 - accurac
Epoch 4/15
21/21 [==============================] - 12s 599ms/step - loss: 0.0830 - accurac
Epoch 5/15
21/21 [==============================] - 12s 597ms/step - loss: 0.0703 - accurac
Epoch 6/15
21/21 [==============================] - 12s 597ms/step - loss: 0.0610 - accurac
Epoch 7/15
21/21 [==============================] - 12s 599ms/step - loss: 0.0557 - accurac
Epoch 8/15
21/21 [==============================] - 12s 595ms/step - loss: 0.0487 - accurac
Epoch 9/15
21/21 [==============================] - 12s 594ms/step - loss: 0.0455 - accurac
Epoch 10/15
21/21 [==============================] - 12s 595ms/step - loss: 0.0433 - accurac
Epoch 11/15
21/21 [==============================] - 12s 595ms/step - loss: 0.0389 - accurac
Epoch 12/15
21/21 [==============================] - 12s 595ms/step - loss: 0.0359 - accurac
Epoch 13/15
21/21 [==============================] - 12s 595ms/step - loss: 0.0342 - accurac
Epoch 14/15
21/21 [==============================] - 12s 595ms/step - loss: 0.0316 - accurac
Epoch 15/15
21/21 [==============================] - 12s 595ms/step - loss: 0.0307 - accurac
> model evaluation
313/313 [==============================] - 2s 6ms/step - loss: 0.0378 - accuracy
[0.037829719483852386, 0.9876000285148621]
```

different optimizer, learning rate

now optimizer with learning rate = 1e-6

```
# 2D,  grösssseres Conv2D eingebaut, mit mehr dense am ende
# andere optimizer function RMSprop, mit learning rate

from keras.optimizers import RMSprop

model = keras.Sequential([
    keras.layers.Conv2D(32, kernel_size=(6, 6),
                activation='relu',
                input_shape=(28,28,1)),
    keras.layers.MaxPooling2D(pool_size=(5, 5)),
    keras.layers.Dropout(0.25),
    keras.layers.Flatten(),
    keras.layers.Dense(255, activation='relu'),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax'),
])

model.compile(optimizer = keras.optimizers.RMSprop(lr=1e-6),
              loss = keras.losses.categorical_crossentropy,
              metrics=['accuracy'])

model.fit(x_train_2d_stan, y_train,
          batch_size = 128,
          epochs = 15,
          #validation_data=(x_test, y_test)),
          validation_split = 0.3,
          shuffle = True,
          verbose = 1)

print('> model evaluation')
model.evaluate(x_test_2d_stan, y_test, verbose = 1)
```

```
Epoch 1/15
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/optimizer_v2/opti
  "The `lr` argument is deprecated, use `learning_rate` instead.")
329/329 [==============================] - 16s 46ms/step - loss: 2.2897 - accura
Epoch 2/15
329/329 [==============================] - 15s 45ms/step - loss: 2.2666 - accura
Epoch 3/15
329/329 [==============================] - 15s 45ms/step - loss: 2.2451 - accura
Epoch 4/15
329/329 [==============================] - 15s 45ms/step - loss: 2.2231 - accura
Epoch 5/15
329/329 [==============================] - 15s 46ms/step - loss: 2.2011 - accura
Epoch 6/15
329/329 [==============================] - 15s 46ms/step - loss: 2.1789 - accura
Epoch 7/15
329/329 [==============================] - 15s 46ms/step - loss: 2.1557 - accura
Epoch 8/15
329/329 [==============================] - 15s 46ms/step - loss: 2.1320 - accura
Epoch 9/15
329/329 [==============================] - 15s 45ms/step - loss: 2.1081 - accura
Epoch 10/15
329/329 [==============================] - 15s 46ms/step - loss: 2.0831 - accura
Epoch 11/15
329/329 [==============================] - 15s 46ms/step - loss: 2.0564 - accura
Epoch 12/15
329/329 [==============================] - 15s 46ms/step - loss: 2.0295 - accura
Epoch 13/15
329/329 [==============================] - 15s 46ms/step - loss: 2.0005 - accura
Epoch 14/15
329/329 [==============================] - 15s 46ms/step - loss: 1.9723 - accura
Epoch 15/15
329/329 [==============================] - 15s 46ms/step - loss: 1.9411 - accura
> model evaluation
```

```
313/313 [==============================] - 2s 6ms/step - loss: 1.8848 - accuracy
[1.884812831878662, 0.6654999852180481]
```

different optimizer, learning rate 2

now optimizer with learning rate = 0.01

```
# 2D,  grösssseres Conv2D eingebaut, mit mehr dense am ende
# andere optimizer function RMSprop, mit learning rate : 0.01

from keras.optimizers import RMSprop

model = keras.Sequential([
    keras.layers.Conv2D(32, kernel_size=(6, 6),
                activation='relu',
                input_shape=(28,28,1)),
    keras.layers.MaxPooling2D(pool_size=(5, 5)),
    keras.layers.Dropout(0.25),
    keras.layers.Flatten(),
    keras.layers.Dense(255, activation='relu'),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax'),
])

model.compile(optimizer = keras.optimizers.RMSprop(lr=0.01),
              loss = keras.losses.categorical_crossentropy,
              metrics=['accuracy'])

model.fit(x_train_2d_stan, y_train,
           batch_size = 128,
           epochs = 15,
           #validation_data=(x_test, y_test)),
           validation_split = 0.3,
           shuffle = True,
           verbose = 1)

print('> model evaluation')
model.evaluate(x_test_2d_stan, y_test, verbose = 1)
```

```
Epoch 1/15
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/optimizer_v2/opti
  "The `lr` argument is deprecated, use `learning_rate` instead.")
329/329 [==============================] - 15s 45ms/step - loss: 0.2048 - accura
Epoch 2/15
329/329 [==============================] - 15s 45ms/step - loss: 0.0708 - accura
Epoch 3/15
329/329 [==============================] - 15s 45ms/step - loss: 0.0674 - accura
```

```
Epoch 4/15
329/329 [==============================] - 15s 44ms/step - loss: 0.0720 - accura
Epoch 5/15
329/329 [==============================] - 15s 44ms/step - loss: 0.0721 - accura
Epoch 6/15
329/329 [==============================] - 15s 44ms/step - loss: 0.0732 - accura
Epoch 7/15
329/329 [==============================] - 15s 45ms/step - loss: 0.0777 - accura
Epoch 8/15
329/329 [==============================] - 15s 44ms/step - loss: 0.0756 - accura
Epoch 9/15
329/329 [==============================] - 15s 45ms/step - loss: 0.0718 - accura
Epoch 10/15
329/329 [==============================] - 15s 44ms/step - loss: 0.0812 - accura
Epoch 11/15
329/329 [==============================] - 15s 44ms/step - loss: 0.0783 - accura
Epoch 12/15
329/329 [==============================] - 15s 45ms/step - loss: 0.0728 - accura
Epoch 13/15
329/329 [==============================] - 15s 44ms/step - loss: 0.0771 - accura
Epoch 14/15
329/329 [==============================] - 15s 44ms/step - loss: 0.0774 - accura
Epoch 15/15
329/329 [==============================] - 15s 45ms/step - loss: 0.0761 - accura
> model evaluation
313/313 [==============================] - 2s 5ms/step - loss: 0.1484 - accuracy
[0.14839889109134674, 0.9807999730110168]
```

different optimizer, learning rate 3

now optimizer with learning rate = 0.1

```
# 2D,  grösssseres Conv2D eingebaut, mit mehr dense am ende
# andere optimizer function RMSprop, mit learning rate : 0.1

from keras.optimizers import RMSprop

model = keras.Sequential([
    keras.layers.Conv2D(32, kernel_size=(6, 6),
                activation='relu',
                input_shape=(28,28,1)),
    keras.layers.MaxPooling2D(pool_size=(5, 5)),
    keras.layers.Dropout(0.25),
    keras.layers.Flatten(),
    keras.layers.Dense(255, activation='relu'),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax'),
])

model.compile(optimizer = keras.optimizers.RMSprop(lr=0.1),
              loss = keras.losses.categorical_crossentropy,
              metrics=['accuracy'])

model.fit(x_train_2d_stan, y_train,
           batch_size = 128,
           epochs = 15,
           #validation_data=(x_test, y_test)),
           validation_split = 0.3,
           shuffle = True,
           verbose = 1)

print('> model evaluation')
model.evaluate(x_test_2d_stan, y_test, verbose = 1)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/optimizer_v2/opti
  "The `lr` argument is deprecated, use `learning_rate` instead.")
Epoch 1/15
329/329 [==============================] - 15s 44ms/step - loss: 68.4931 - accur
Epoch 2/15
329/329 [==============================] - 14s 44ms/step - loss: 2.3094 - accura
Epoch 3/15
329/329 [==============================] - 15s 44ms/step - loss: 2.3102 - accura
Epoch 4/15
329/329 [==============================] - 14s 44ms/step - loss: 2.3097 - accura
Epoch 5/15
329/329 [==============================] - 15s 44ms/step - loss: 2.3100 - accura
Epoch 6/15
329/329 [==============================] - 14s 44ms/step - loss: 2.3098 - accura
Epoch 7/15
329/329 [==============================] - 14s 44ms/step - loss: 2.3096 - accura
Epoch 8/15
329/329 [==============================] - 15s 44ms/step - loss: 2.3097 - accura
Epoch 9/15
329/329 [==============================] - 14s 44ms/step - loss: 2.3099 - accura
Epoch 10/15
329/329 [==============================] - 14s 44ms/step - loss: 2.3096 - accura
Epoch 11/15
329/329 [==============================] - 14s 44ms/step - loss: 2.3097 - accura
Epoch 12/15
329/329 [==============================] - 14s 44ms/step - loss: 2.3091 - accura
Epoch 13/15
329/329 [==============================] - 15s 44ms/step - loss: 2.3096 - accura
Epoch 14/15
329/329 [==============================] - 15s 44ms/step - loss: 2.3095 - accura
Epoch 15/15
329/329 [==============================] - 15s 44ms/step - loss: 2.3098 - accura
> model evaluation
313/313 [==============================] - 2s 5ms/step - loss: 2.3271 - accuracy
[2.3270788192749023, 0.10000000149011612]
```

Prediction accuracy

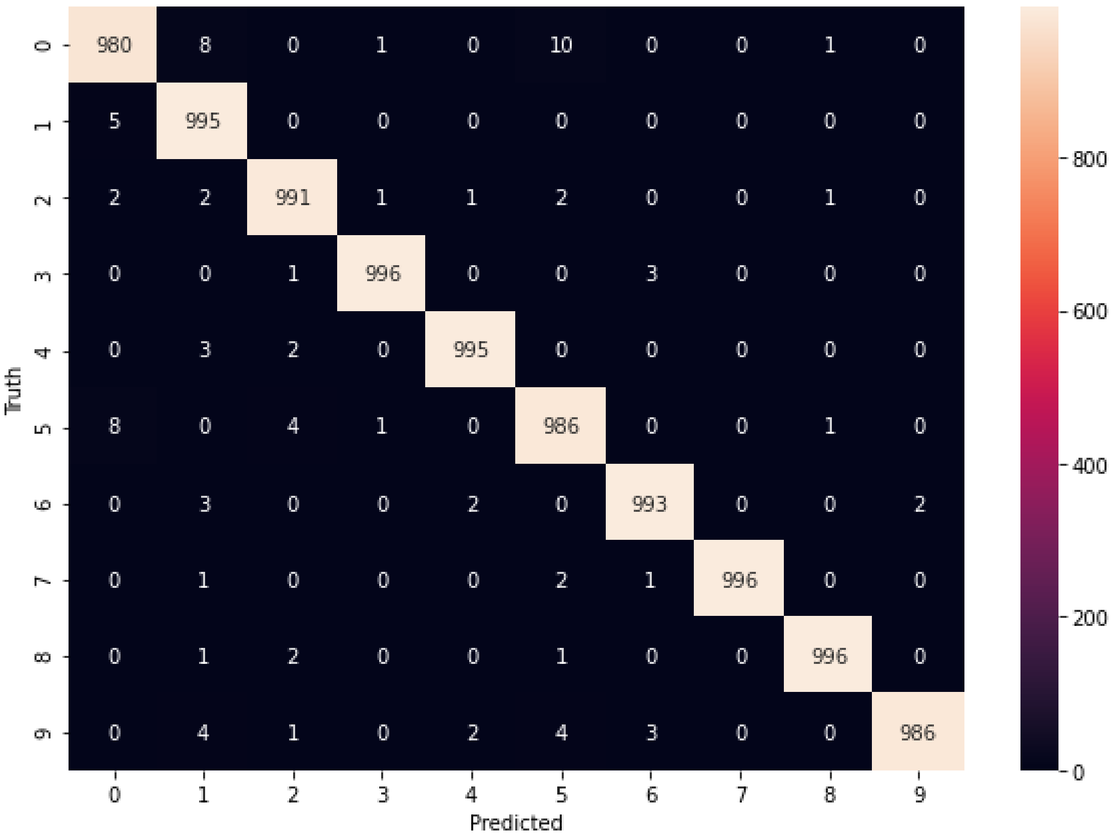
```
y_test_labels = [np.argmax(i) for i in y_test]
```

```
    [0, 1, 2, 3, 4]
```

```
y_predictedc3 = modelc3.predict(x_test_2d_stan)
y_predicted_labelsc3 = [np.argmax(i) for i in y_predictedc3]
conf_mc3 = tf.math.confusion_matrix(labels = y_test_labels, predictions = y_predicted_

plt.figure(figsize = (10,7))
sn.heatmap(conf_mc3, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

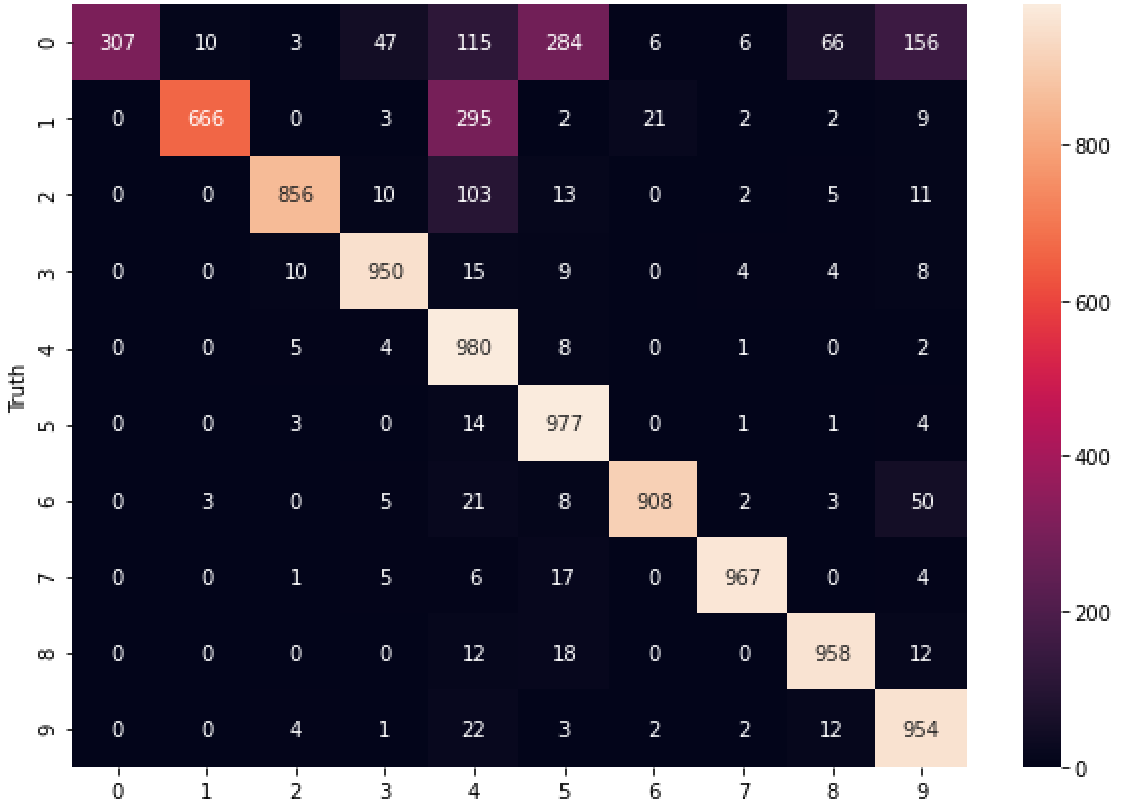
```
    Text(69.0, 0.5, 'Truth')
```



```
y_predicted1 = model1.predict(x_test_stan)
y_predicted_labels1 = [np.argmax(i) for i in y_predicted1]
conf_m1 = tf.math.confusion_matrix(labels = y_test_labels, predictions = y_predicted_l

plt.figure(figsize = (10,7))
sn.heatmap(conf_m1, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(69.0, 0.5, 'Truth')
```



Truth	0	1	2	3	4	5	6	7	8	9
0	307	10	3	47	115	284	6	6	66	156
1	0	666	0	3	295	2	21	2	2	9
2	0	0	856	10	103	13	0	2	5	11
3	0	0	10	950	15	9	0	4	4	8
4	0	0	5	4	980	8	0	1	0	2
5	0	0	3	0	14	977	0	1	1	4
6	0	3	0	5	21	8	908	2	3	50
7	0	0	1	5	6	17	0	967	0	4
8	0	0	0	0	12	18	0	0	958	12
9	0	0	4	1	22	3	2	2	12	954

Network Comparison

```
model = keras.Sequential([
    keras.layers.Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=(28,28,1)),
    keras.layers.Conv2D(64, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Dropout(0.25),
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(num_classes, activation='softmax'),
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train_2d_stan, y_train,
          batch_size = 128,
          epochs = 10,
          #validation_data=(x_test, y_test)),
          validation_split = 0.3,
          shuffle = True,
          verbose = 1)
```

```
print('> model evaluation')
model.evaluate(x_test_2d_stan, y_test, verbose = 1)
```

```
Epoch 1/10
329/329 [==============================] - 80s 241ms/step - loss: 0.1647 - accur
Epoch 2/10
329/329 [==============================] - 79s 240ms/step - loss: 0.0492 - accur
Epoch 3/10
329/329 [==============================] - 79s 240ms/step - loss: 0.0329 - accur
Epoch 4/10
329/329 [==============================] - 79s 240ms/step - loss: 0.0292 - accur
Epoch 5/10
329/329 [==============================] - 79s 239ms/step - loss: 0.0234 - accur
Epoch 6/10
329/329 [==============================] - 79s 240ms/step - loss: 0.0213 - accur
Epoch 7/10
329/329 [==============================] - 79s 239ms/step - loss: 0.0186 - accur
Epoch 8/10
329/329 [==============================] - 79s 239ms/step - loss: 0.0180 - accur
Epoch 9/10
329/329 [==============================] - 79s 239ms/step - loss: 0.0156 - accur
Epoch 10/10
329/329 [==============================] - 79s 239ms/step - loss: 0.0143 - accur
> model evaluation
313/313 [==============================] - 5s 17ms/step - loss: 0.0411 - accurac
[0.04109307751059532, 0.9897000193595886]
```

✓ 13 min 13 s Abgeschlossen um 17:09 ● ✕