2_deep_learning_keras

May 28, 2024

1 Neural Network with Keras

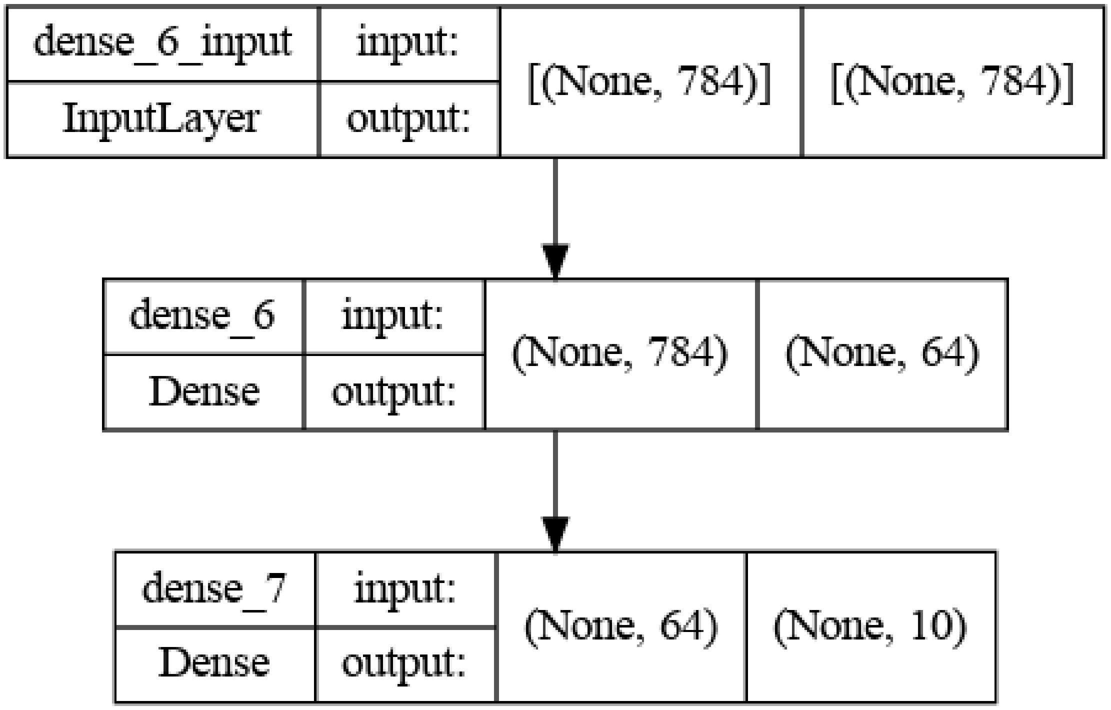
We have made a lot of effort to program our neural network that is able to classify differenr handwritten number with the help of numpy. A lot of other people did that already and since this is the basis for many applications nowadays, a large number of API (application programming interfaces) exist. Python plays therby a leading role. We will use in the follwing the interface provided by the `keras` module. `keras` is actually sitting on top of the real machine learning API, which is in our case `tensorflow`. `keras` makes the use of tensorflow a bit more friendly and from the example below, you wil recognize by how much shorter our code gets with the keras and tensorflow API.

1.1 MNIST Data Set (Keras)

This loads the same data as in our previous notebook, except that the function to do that is directly provided by `keras`.

1.2 Build the model

The next few lines create the whole neural network with an input layer, a hidden layer with 64 neurons and and output layer with 10 neurons.



1.3 Compile the model

The `compile` method assembles everything to create a model for training. You can specify here the stochastic gradient descent method in the same way as the loss function.

1.4 Train the model

Finally, the `fit` method allows us to train the model for a specified number of epochs.

```
Epoch 1/20
 275/1875 [===>…] - ETA: 0s - loss: 0.3216 - accuracy:
0.9075

2023-07-11 13:53:05.833950: W
tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 188160000
exceeds 10% of free system memory.

1875/1875 [==============================] - 1s 567us/step - loss: 0.3172 -
accuracy: 0.9098
Epoch 2/20
1875/1875 [==============================] - 1s 552us/step - loss: 0.3084 -
accuracy: 0.9119
Epoch 3/20
1875/1875 [==============================] - 1s 549us/step - loss: 0.3006 -
accuracy: 0.9144
Epoch 4/20
```

```
1875/1875 [==============================] - 1s 551us/step - loss: 0.2935 -
accuracy: 0.9169
Epoch 5/20
1875/1875 [==============================] - 1s 555us/step - loss: 0.2870 -
accuracy: 0.9188
Epoch 6/20
1875/1875 [==============================] - 1s 553us/step - loss: 0.2808 -
accuracy: 0.9201
Epoch 7/20
1875/1875 [==============================] - 1s 551us/step - loss: 0.2753 -
accuracy: 0.9218
Epoch 8/20
1875/1875 [==============================] - 1s 549us/step - loss: 0.2700 -
accuracy: 0.9233
Epoch 9/20
1875/1875 [==============================] - 1s 550us/step - loss: 0.2649 -
accuracy: 0.9247
Epoch 10/20
1875/1875 [==============================] - 1s 550us/step - loss: 0.2601 -
accuracy: 0.9265
Epoch 11/20
1875/1875 [==============================] - 1s 566us/step - loss: 0.2556 -
accuracy: 0.9278
Epoch 12/20
1875/1875 [==============================] - 1s 546us/step - loss: 0.2512 -
accuracy: 0.9288
Epoch 13/20
1875/1875 [==============================] - 1s 548us/step - loss: 0.2469 -
accuracy: 0.9302
Epoch 14/20
1875/1875 [==============================] - 1s 550us/step - loss: 0.2429 -
accuracy: 0.9314
Epoch 15/20
1875/1875 [==============================] - 1s 562us/step - loss: 0.2390 -
accuracy: 0.9323
Epoch 16/20
1875/1875 [==============================] - 1s 549us/step - loss: 0.2352 -
accuracy: 0.9333
Epoch 17/20
1875/1875 [==============================] - 1s 546us/step - loss: 0.2316 -
accuracy: 0.9341
Epoch 18/20
1875/1875 [==============================] - 1s 547us/step - loss: 0.2280 -
accuracy: 0.9352
Epoch 19/20
1875/1875 [==============================] - 1s 546us/step - loss: 0.2247 -
accuracy: 0.9361
Epoch 20/20
```

```
1875/1875 [==============================] - 1s 548us/step - loss: 0.2214 -
accuracy: 0.9371
```

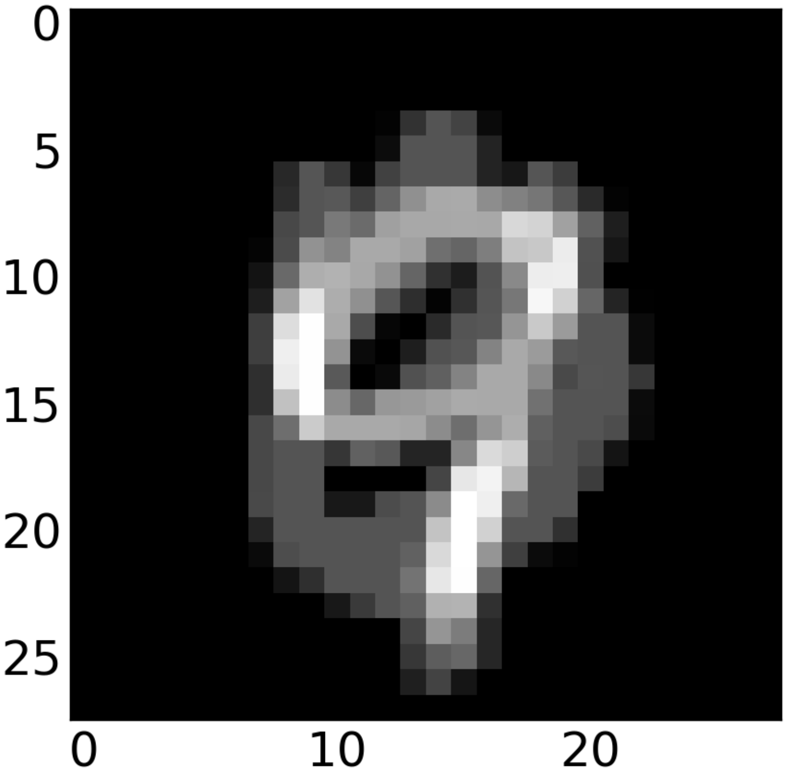
```
<keras.callbacks.History at 0x7f54450047c0>
```

1.5 Testing the model

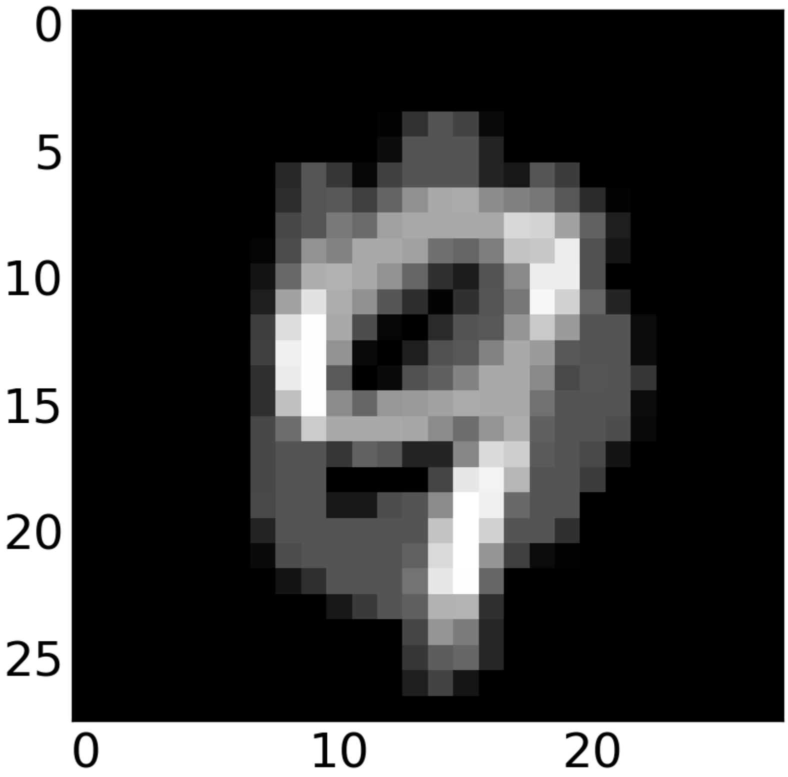
We may now use our trained model to predict the number in the image with the `model.predict` function. This delivers an array of 10 numbers, which represent the confidences that the number $0, \ldots, 9$ are contained. The index of the biggest number thus represents the number contained in the image.

```
array([[3.3470042e-05, 8.6805121e-06, 1.1352806e-04, 5.8598683e-04,
        1.9212976e-02, 8.8488462e-04, 1.8833722e-05, 1.4240003e-02,
        5.1060988e-04, 9.6439105e-01]], dtype=float32)
```

```
<matplotlib.image.AxesImage at 0x7f5446a01fd0>
```

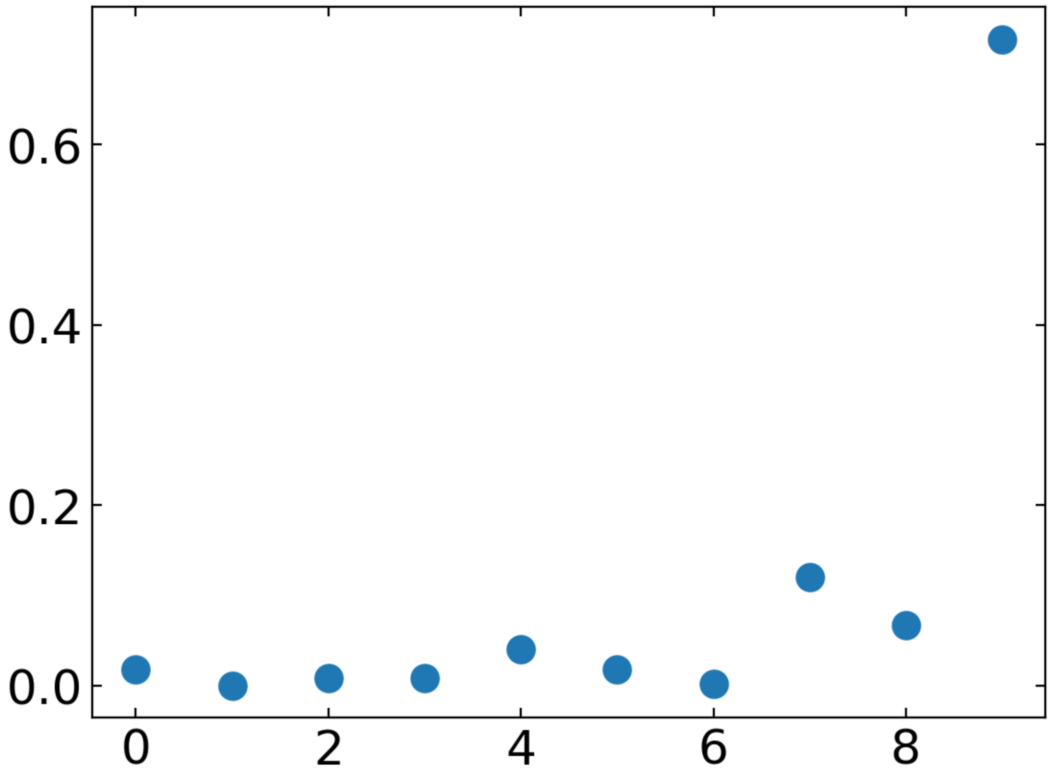


```
prediction:  9
```



```
array([[1.77512094e-02, 9.22485924e-05, 8.42965860e-03, 8.65467917e-03,
        4.02621441e-02, 1.75370630e-02, 1.91166031e-03, 1.20765775e-01,
        6.74873590e-02, 7.17108250e-01]], dtype=float32)
```

```
<matplotlib.collections.PathCollection at 0x7f544685a640>
```


0.6
0.4
0.2
0.0
0
2
4
6
8