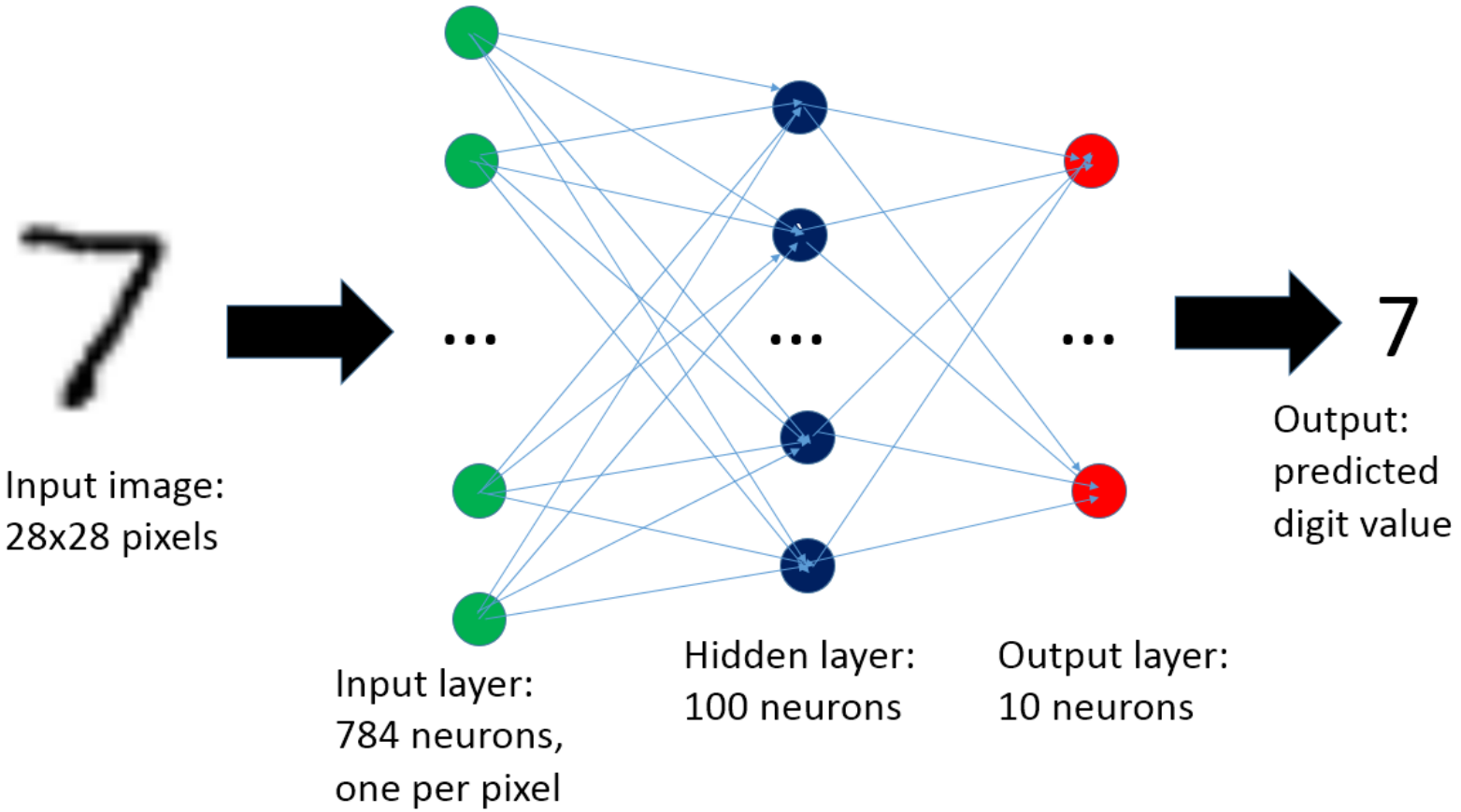
MNIST number recognition using simple neural network

Oleksandra Panova

Taras Shevchenko National University of Kiev

September 9, 2019

Simple neural network with 1 hidden layer



Mathematical model of neuron:

$$x_k^{i+1} = f\left(\sum_{j=1}^{N} w_j^k x_j^i\right)$$

Import all necessary packages

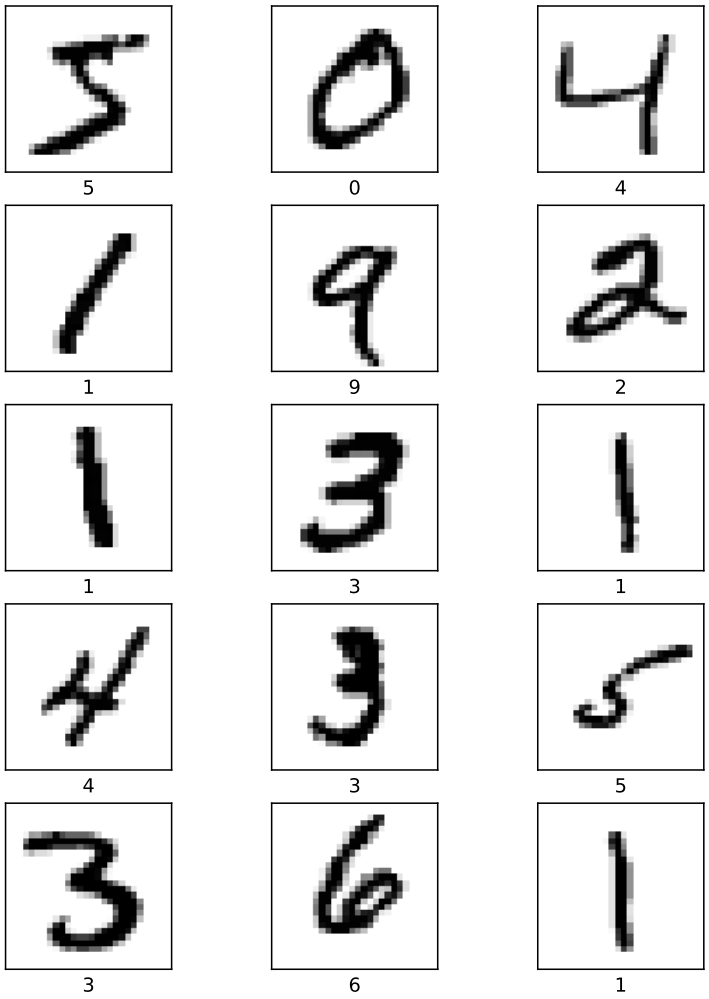
```
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
import numpy as np
import matplotlib.pyplot as plt
```

Load and normalize data

```
(train_images, train_labels), (test_images, test_labels) =
    mnist.load_data()

train_images = train_images / 255.0
test_images = test_images / 255.0
```

MNIST dataset




```
plt.figure(figsize=(7,9))
for i in range(15):
    plt.subplot(5,3,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(train_images[i],
    cmap=plt.cm.binary)
    plt.xlabel(train_labels[i])
plt.savefig('mnist.pdf')
plt.show()
```

Create and compile model

```
def baseline_model():
    model = Sequential()
    model.add(Flatten(input_shape=(28, 28)))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    model.compile(loss='sparse_categorical_crossentropy',
                  optimizer='adam', metrics=['accuracy'])
    return model
```

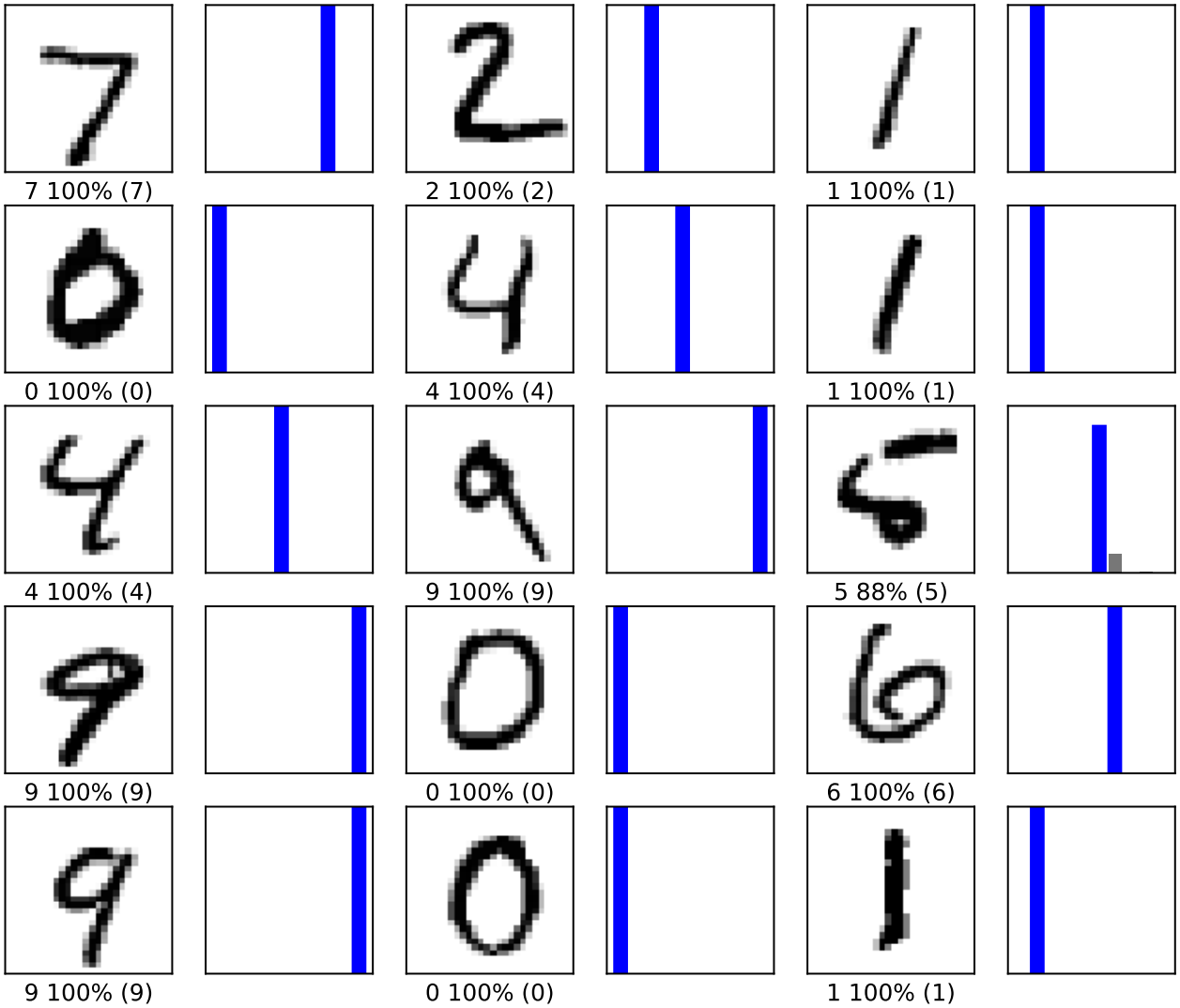
Build and fit model

```
model = baseline_model()
model.fit(train_images, train_labels, epochs=10)
```

Accuracy estimation

```
test_loss, test_acc = model.evaluate(test_images,
    test_labels)
print('Accuracy:', test_acc)
```

Results


7 100% (7)
2 100% (2)
1 100% (1)
0 100% (0)
4 100% (4)
1 100% (1)
4 100% (4)
9 100% (9)
5 88% (5)
9 100% (9)
0 100% (0)
6 100% (6)
9 100% (9)
0 100% (0)
1 100% (1)

Wrong results