DHBW Lörrach
Duale Hochschule Baden-Württemberg

Handgeschriebene Zifferererkennung

Einführung in die künstliche Intelligenz

WWI19A

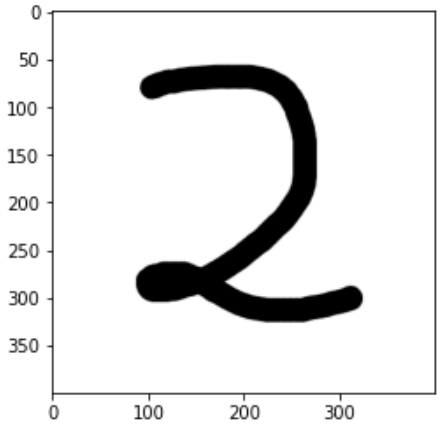
Benedikt Kupfer, Jonas Schlachter, Julian Ortlieb, Patrick Egen

Gliederung

- Einleitung
- Datensatz
- Neuronales Netz
- GUI
- Demo
- Fazit / Verbesserung

Einleitung

- **Aufgabe:** Handgeschrieben Ziffern erkennen
- Durch Algorithmus nicht lösbar
- „Hello World“ der KI



- **Rahmendaten**
  - GUI zur Eingabe mit Maus
  - Ziffer interpretieren und ausgeben
  - Python oder Java

https://logos-download.com/9988-python-logo-download.html

MNIST-Datensatz

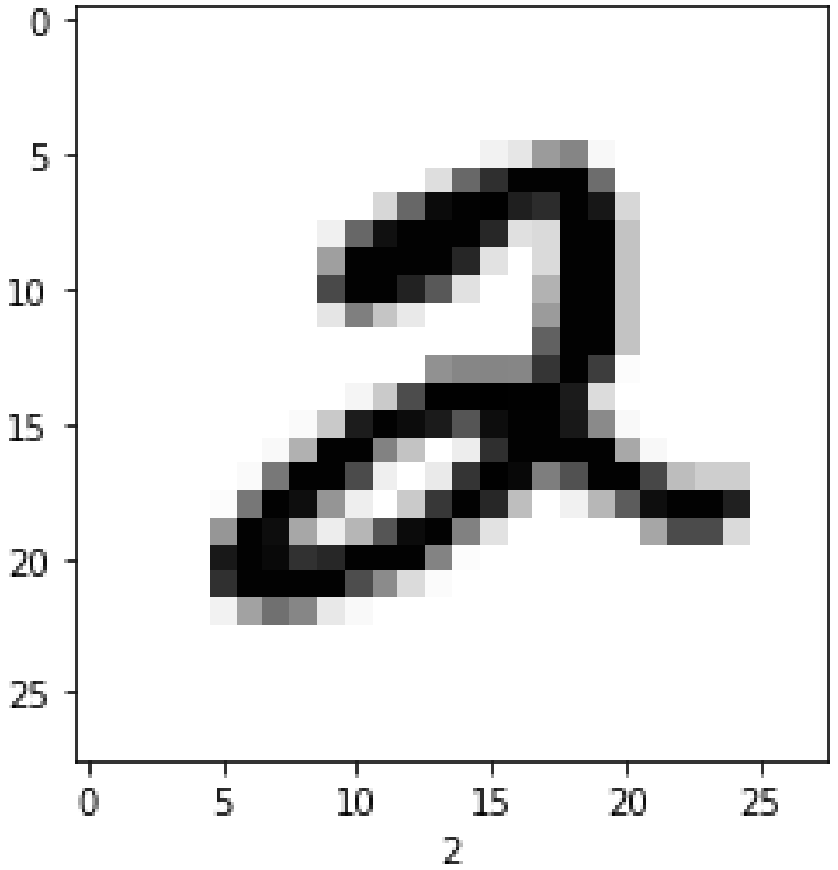
- MNIST-Datensatz
  = *Modified National Institute of Standards and Technology database*
- Datenbank von handgeschriebenen Ziffern
- 60.000 Trainingsdaten
- 10.000 Testdaten

https://upload.wikimedia.org/wikipedia/commons/2/27/MnistExamples.png

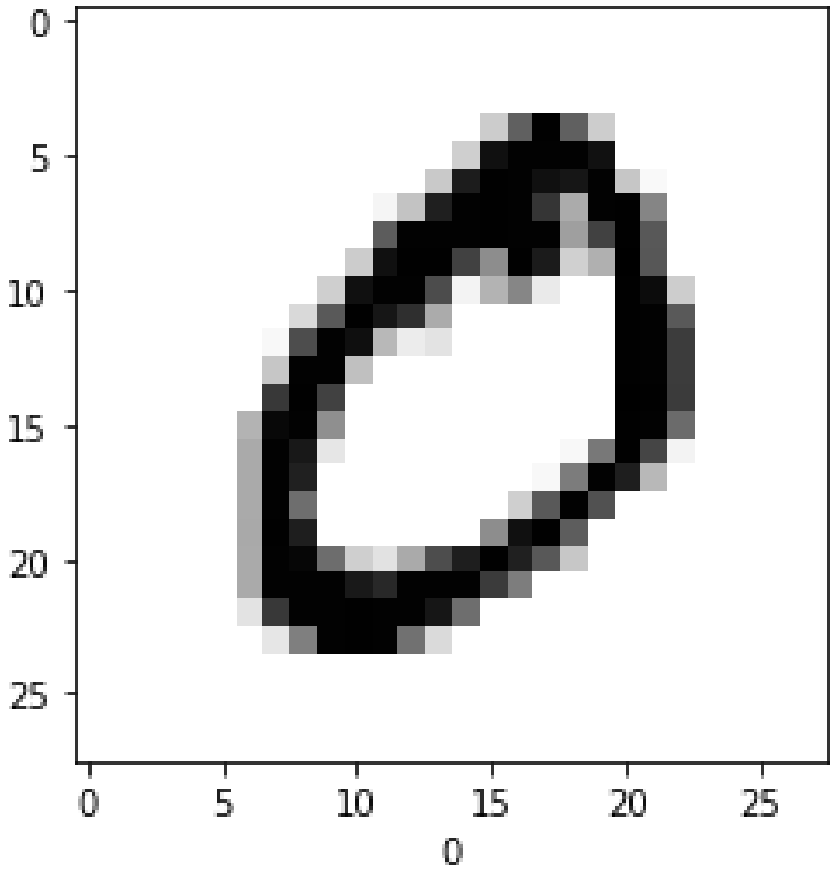
https://www.nist.gov/itl/products-and-services/emnist-dataset

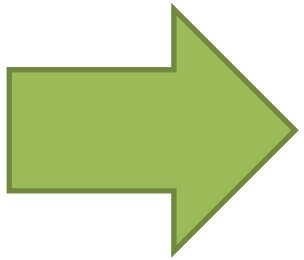
Ziffer

- Ziffer besteht aus 28x28 Pixel
- Schwarzweiß
- Werte von 0 bis 255
- 0 für weiß
- 255 für schwarz



Ziffer als Array



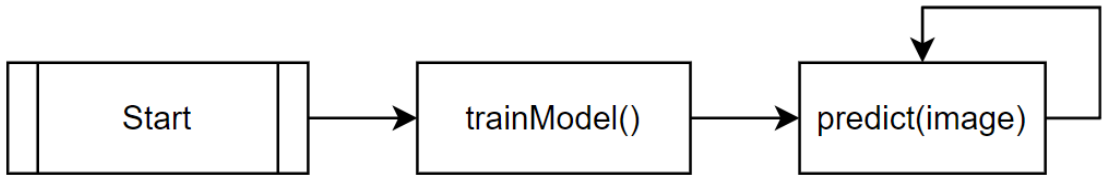


```
[
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 51 159 253 159 50 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 48 238 252 252 252 237 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 54 227 253 252 239 233 252 57 6 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 60 224 252 253 252 202 84 252 253 122 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 163 252 252 252 253 252 252 96 189 253 167 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 51 238 253 253 190 114 253 228 47 79 255 168 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 48 238 252 252 179 12 75 121 21 0 0 253 243 50 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 38 165 253 233 208 84 0 0 0 0 0 0 253 252 165 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 7 178 252 240 71 19 28 0 0 0 0 0 0 253 252 195 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 57 252 252 63 0 0 0 0 0 0 0 0 0 253 252 195 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 198 253 190 0 0 0 0 0 0 0 0 0 0 255 253 196 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 76 246 252 112 0 0 0 0 0 0 0 0 0 0 253 252 148 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 85 252 230 25 0 0 0 0 0 0 0 0 7 135 253 186 12 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 85 252 223 0 0 0 0 0 0 0 0 0 7 131 252 225 71 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 85 252 145 0 0 0 0 0 0 0 0 48 165 252 173 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 86 253 225 0 0 0 0 0 0 0 114 238 253 162 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 85 252 249 146 48 29 85 178 225 253 223 167 56 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 85 252 252 252 229 215 252 252 252 196 130 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 28 199 252 252 253 252 252 233 145 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 25 128 252 253 252 141 37 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
]
```

Neuronales Netz

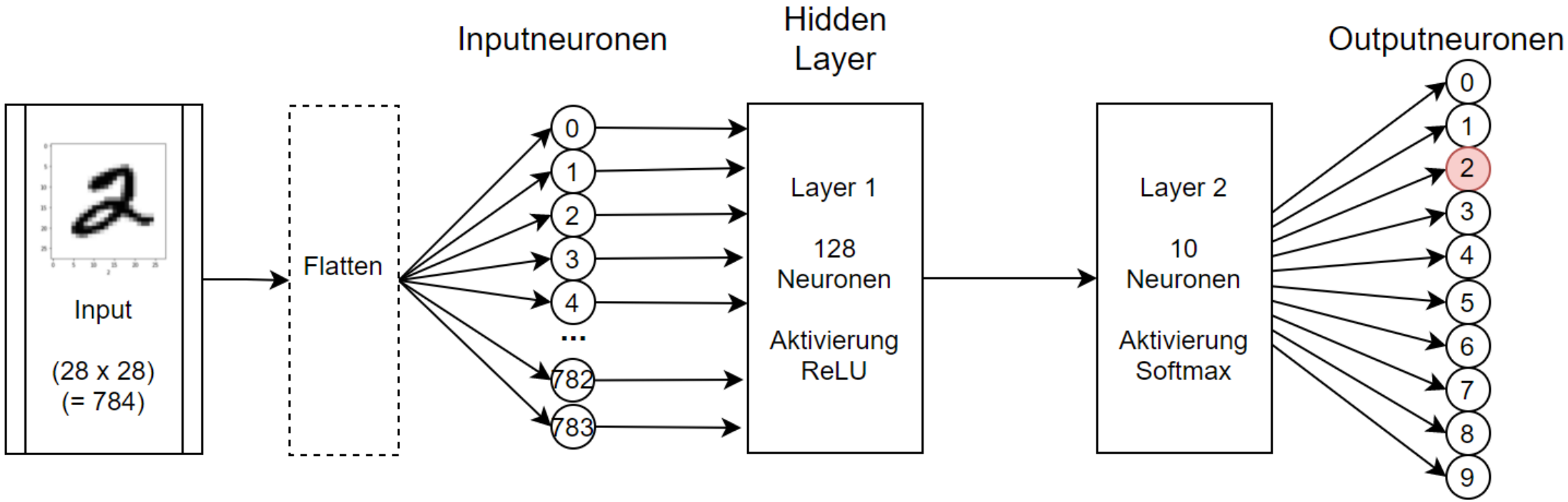
TensorFlow

- Basiert auf Keras & TensorFlow
- Frameworks für neuronale Netze & Machine Learning
- Modell wird einmalig mit MNIST trainiert



- Einfaches & komplexes Model

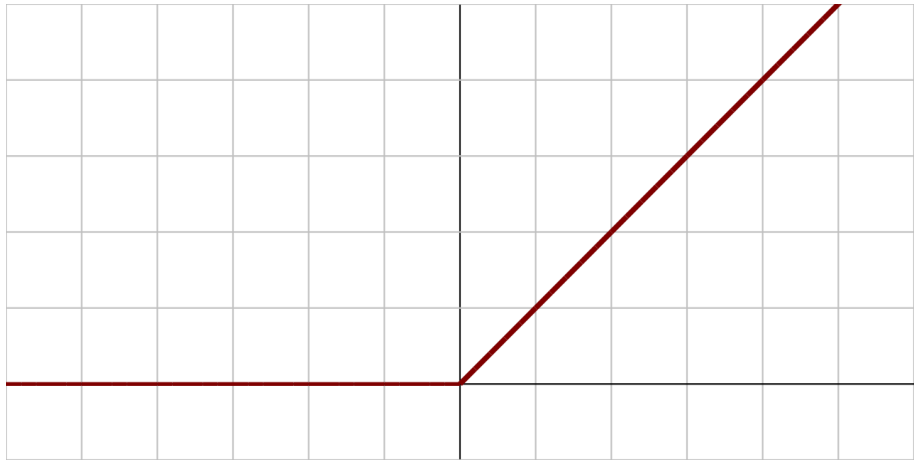
Schichten


Input
(28 x 28)
(= 784)
Flatten
Inputneuronen
0
1
2
3
4
...
782
783
Hidden Layer
Layer 1
128 Neuronen
Aktivierung ReLU
Layer 2
10 Neuronen
Aktivierung Softmax
Outputneuronen
0
1
2
3
4
5
6
7
8
9

Aktivierungsfunktionen

- Layer 1: **ReLU**
  - rectified linear activation function

$$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$
$$= \max\{0, x\} = x\mathbf{1}_{x>0}$$



- Layer 2: **Softmax**
  - Summe aller Ausgaben ergibt 1

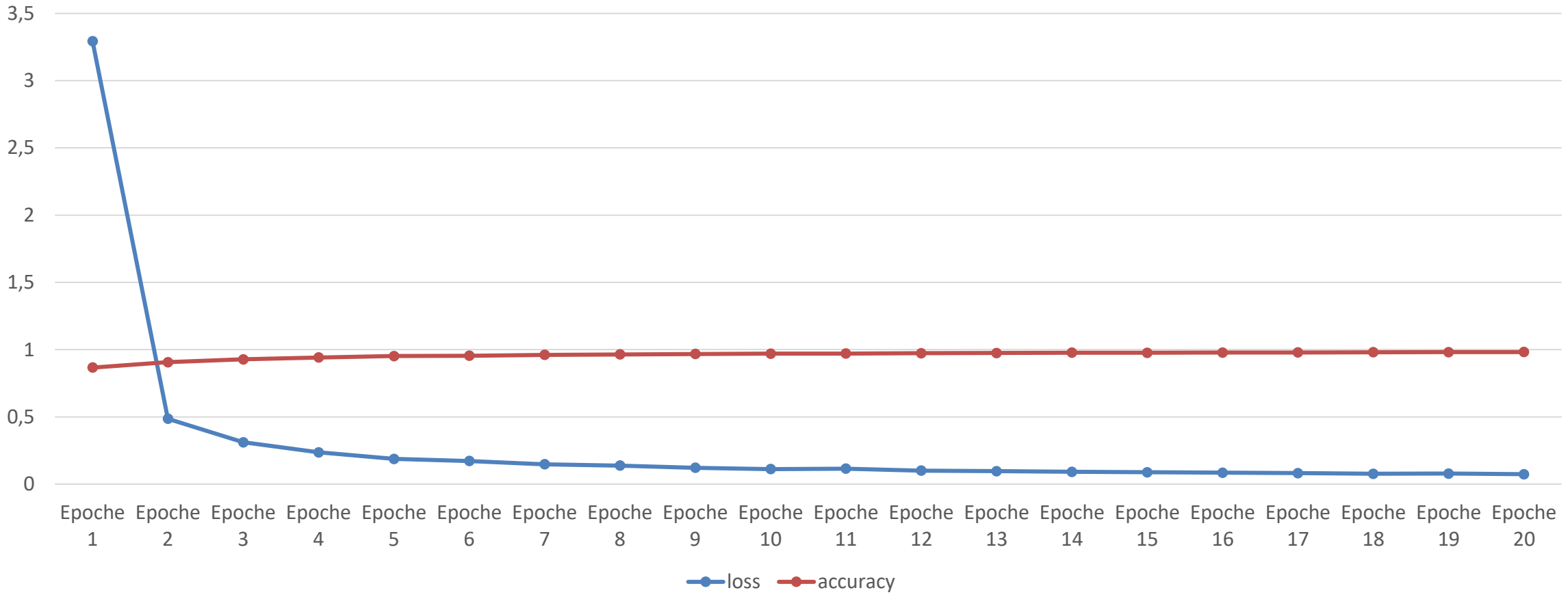
$$\frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}$$

Training

- Optimiert nach Adam-Algorithmus
- Lernrate von 0,0005
- Trainiert über 20 Epochen

- Erste Experimente:
  - Lernrate 0,01
  - Trainiert über 10 Epochen
  - Ergebnisse nicht zufriedenstellend

Training einfaches Modell – Genauigkeit: 96%

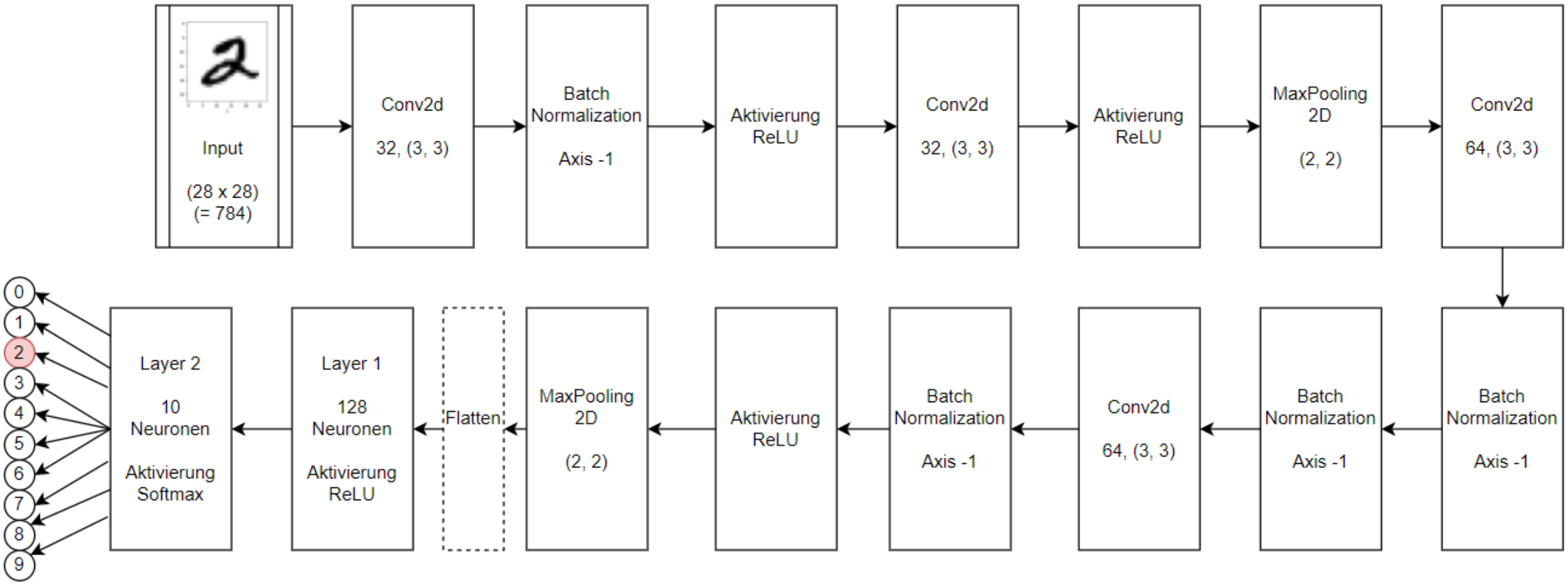


Code-Beispiel: TensorFlow

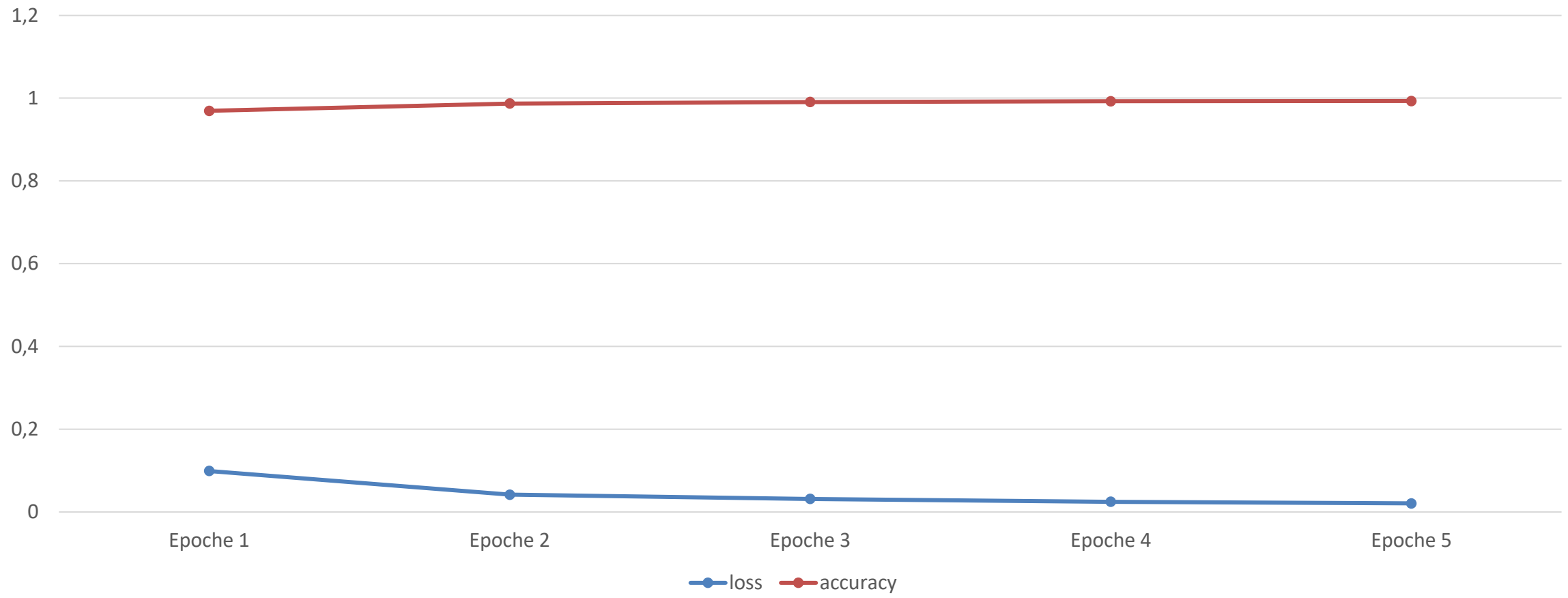
```python
def trainModel():
    mnist = tf.keras.datasets.mnist
    (train_images, train_labels), (test_images, test_labels) = mnist.load_data()
    model = tf.keras.Sequential([
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation=tf.nn.relu),
        tf.keras.layers.Dense(10, activation=tf.nn.softmax)
    ])

    # compile
    model.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=0.0005),
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'],
    )

    # Feed the model
    model.fit(train_images, train_labels, epochs=20)
    model.save(os.path.join(Path(__file__).resolve().parent, 'model'))
```

```python
    predict = model.predict(img_array)
```

Komplexes Modell



Training erweitertes Modell – Genauigkeit: 99%



GUI

- Webanwendung
- Basiert auf Django
- Nativer HTML, CSS und JS Code
- Development Webserver

https://www.djangoproject.com/

Zahleneingabe per Maus über das UI im Frontend

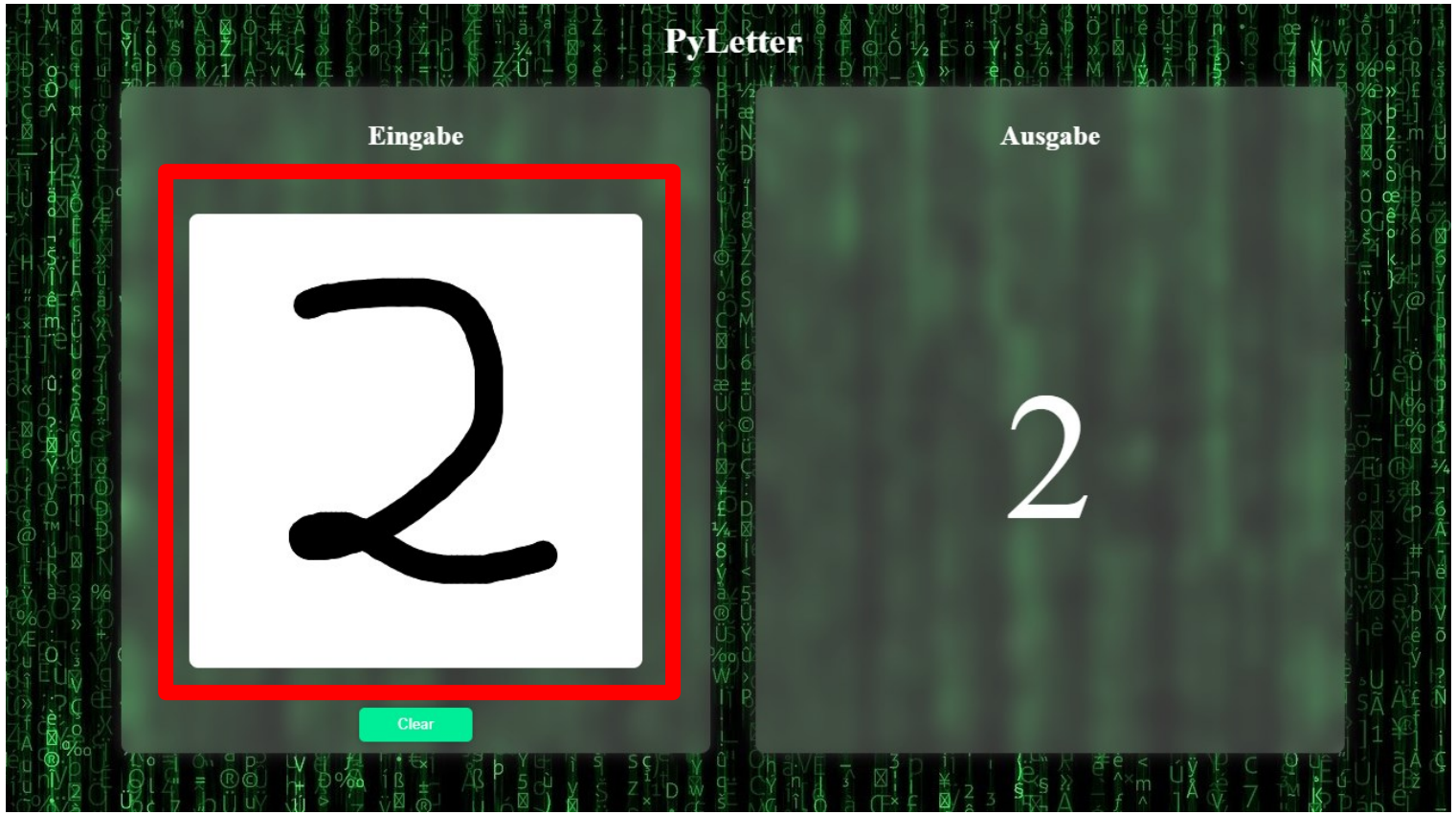
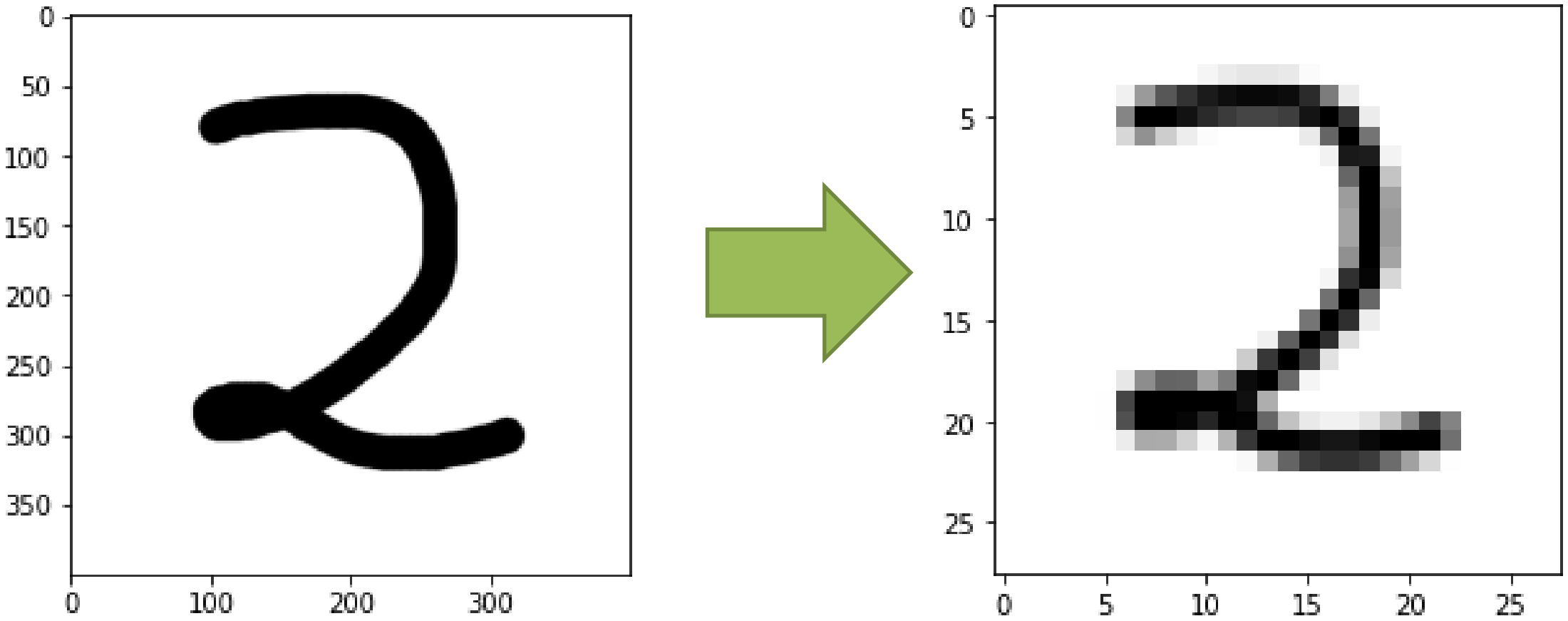
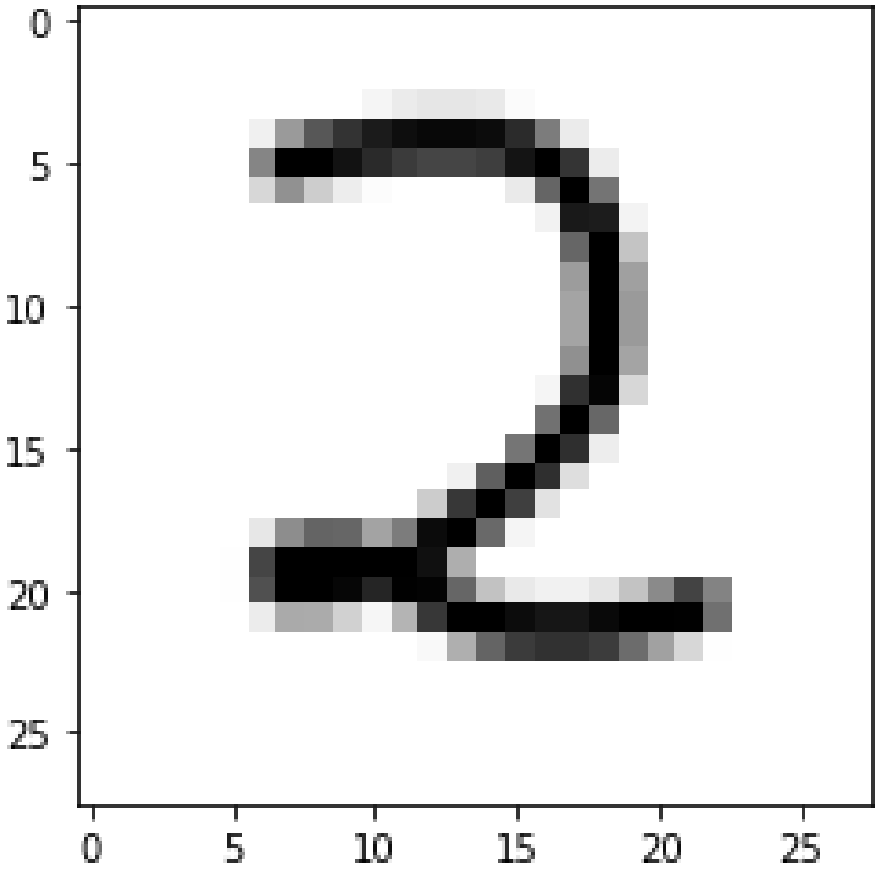
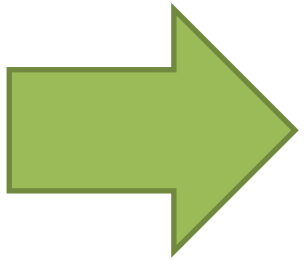


Bild an das Backend senden und verkleinern



Ziffer als Array

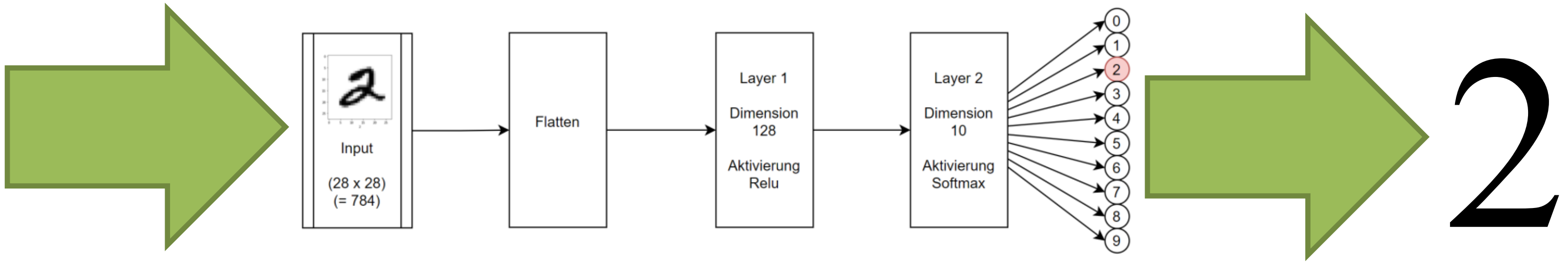




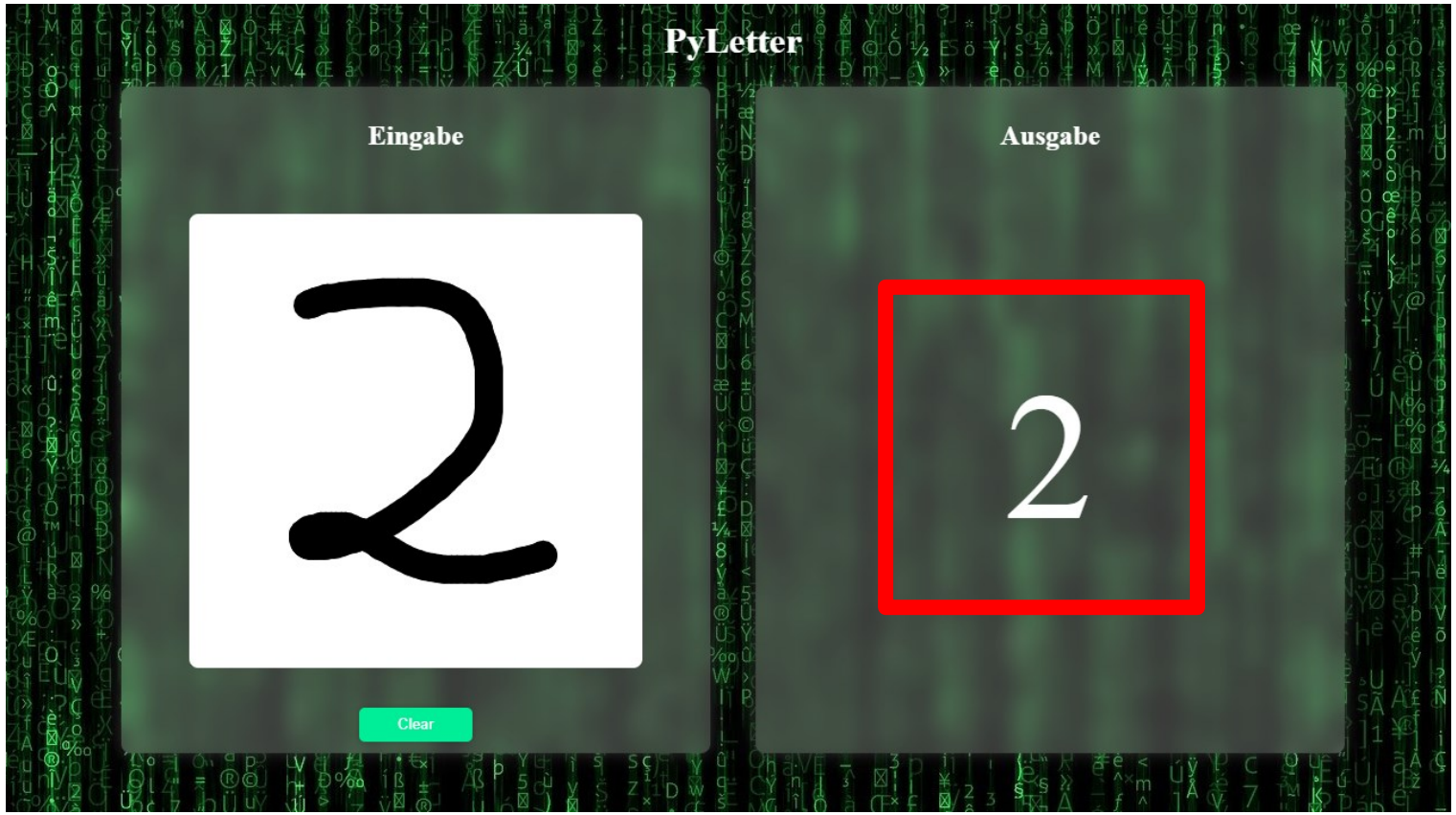
```
[
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 10 20 25 25 22 4 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 15 101 168 204 228 241 247 247 243 212 131 20 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 122 255 255 237 213 196 186 186 193 235 255 203 19 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 40 110 50 18 3 0 0 0 0 21 154 255 139 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 13 230 227 12 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 153 255 59 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 99 255 95 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 91 255 101 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 91 255 101 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 111 255 91 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 207 250 40 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 142 255 152 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 138 255 208 18 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 15 160 255 208 33 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 51 200 255 192 29 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 24 114 155 153 92 131 244 255 150 10 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 1 186 255 255 255 255 255 239 81 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 1 175 255 255 249 218 254 252 153 61 22 14 14 26 60 117 188 124 0 0 0 0]
 [ 0 0 0 0 0 0 19 85 84 46 9 75 200 255 255 243 233 233 246 255 255 254 143 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 6 80 155 196 206 206 195 147 94 40 1 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
]
```

Predict

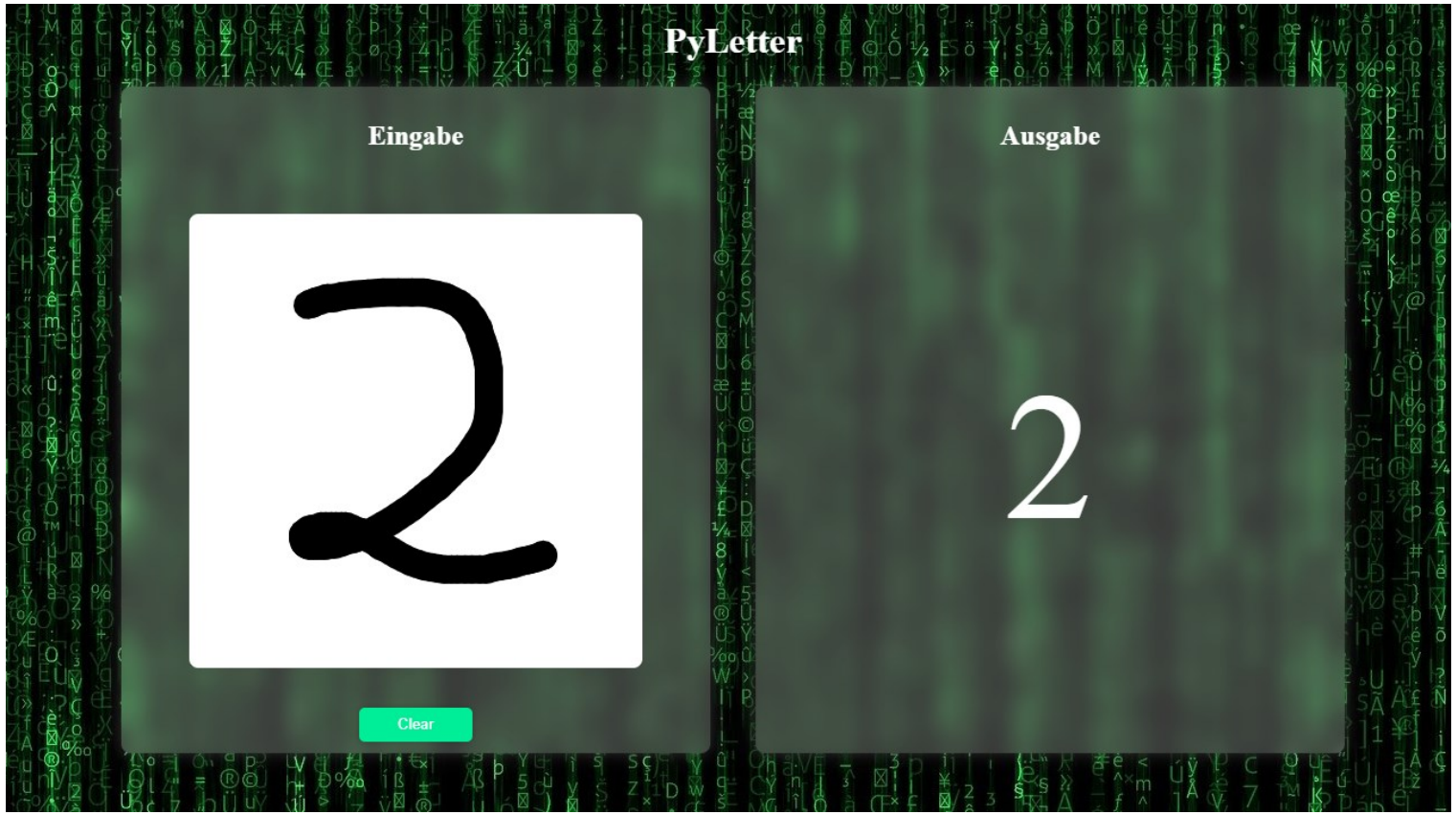
```
[
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 10 20 25 25 22 4 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 15 101 168 204 228 241 247 247 243 212 131 20 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 122 255 255 237 213 196 186 186 193 235 255 203 19 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 40 110 50 18 3 0 0 0 0 21 154 255 139 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 13 230 227 12 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 153 255 59 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 99 255 95 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 91 255 101 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 91 255 101 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 111 255 91 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 207 250 40 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 142 255 152 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 138 255 208 18 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 15 160 255 208 33 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 51 200 255 192 29 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 24 114 155 153 92 131 244 255 150 10 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 1 186 255 255 255 255 255 239 81 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 1 175 255 255 249 218 254 252 153 61 22 14 14 26 60 117 188 124 0 0 0 0 0]
 [ 0 0 0 0 0 0 19 85 84 46 9 75 200 255 255 243 233 233 246 255 255 254 143 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 6 80 155 196 206 206 195 147 94 40 1 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
]
```



Ergebnis an das Frontend senden



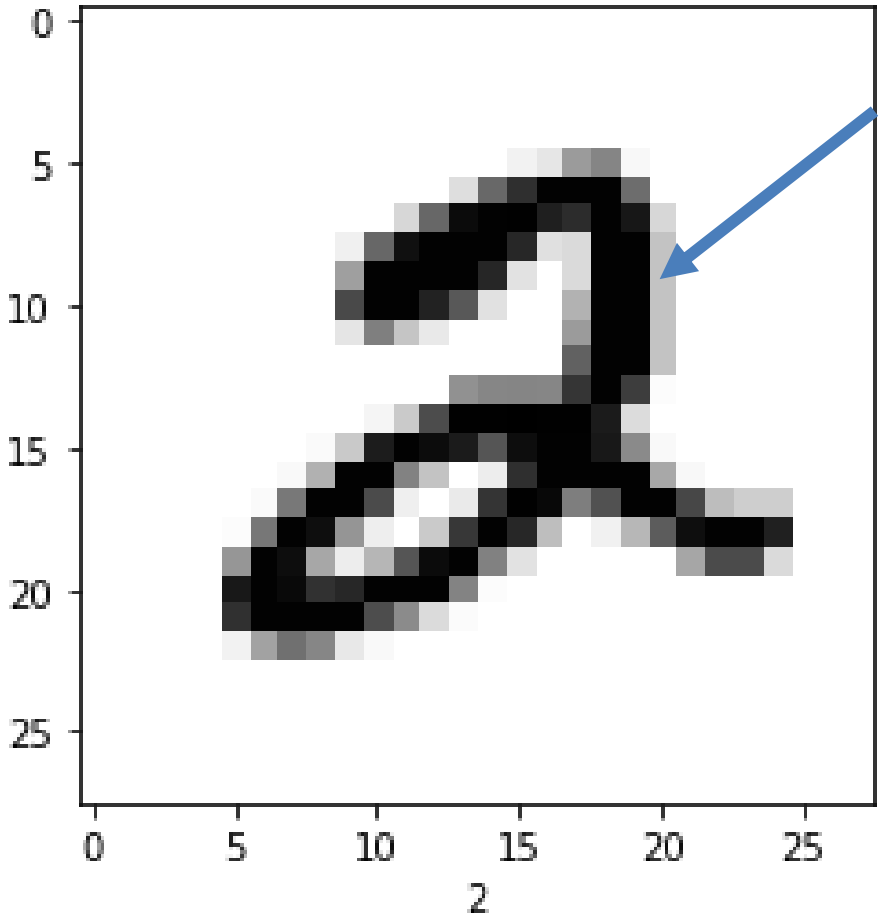
Demo

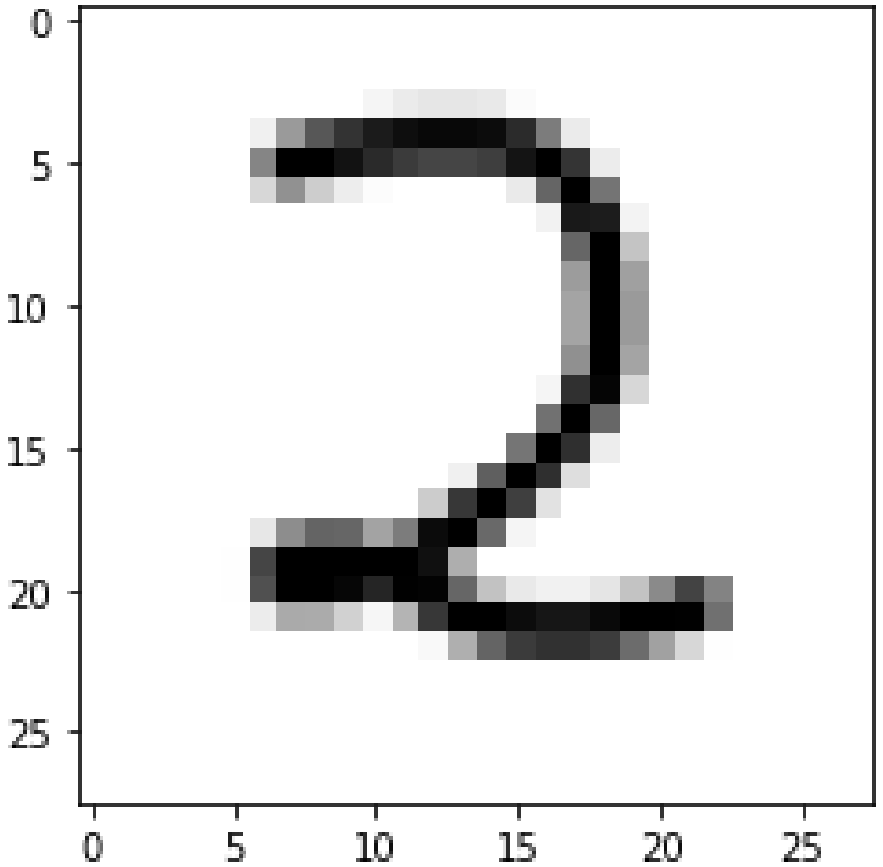


Fazit

- Kritische Zahlen: 0, 6, 9
- Durch das erweiterte Modell und Verwendung von Layern, die für die Bildverarbeitung gemacht wurden steigt die Accuracy enorm

Vergleich





Mögliche Ursachen für Fehler

- Die Art der Trainingsdaten unterscheiden sich von unserer Eingabe
  - Druckstärke des Pinsels
  - Größe des Pinsels
  - Zeichnen mit der Maus
- Unterschiedliche Größen beim Zeichnen
- Fehler im neuronalem Netz können nicht ausgeschlossen werden

Vielen Dank für die Aufmerksamkeit

https://github.com/julianortlieb/pyletter