Laboratory Work 2 Report

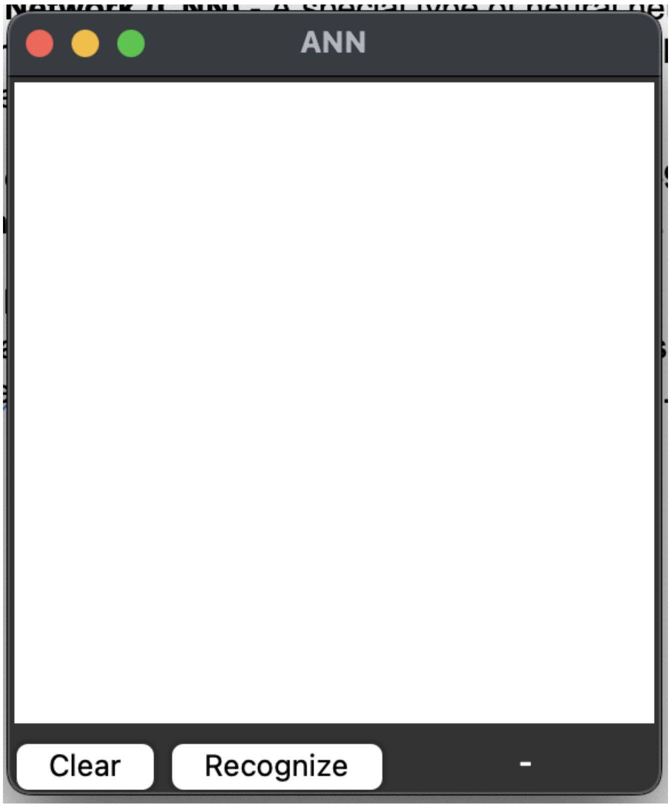
Handwritten Digit Recognition using Neural Networks

Neural Network (ANN) - A mathematical model that mimics how the brain works. It consists of layers of "neurons" that process input data (image pixels) and produce output (which digit it is).

Convolutional Neural Network (CNN) - A special type of neural network designed specifically for image processing. It uses "filters" to detect patterns like edges and shapes, making it more accurate for image recognition.

MNIST Dataset - A collection of 70,000 handwritten digit images (0-9), each 28×28 pixels, used as a standard benchmark for testing machine learning models.

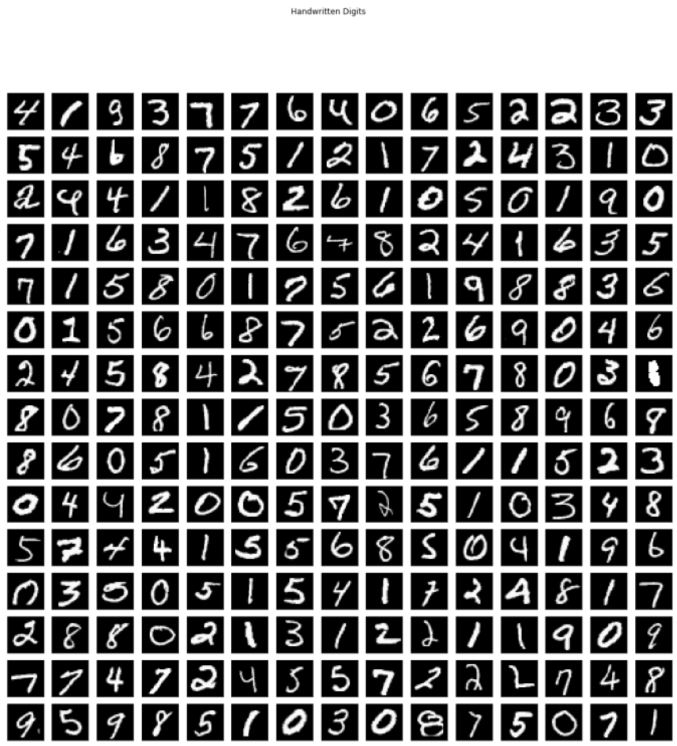
Firstly, created inputField, a tkinter window to allow the drawing and return it as an image. It contains 2 buttons: Clear and Recognize. The label with the result is right next to them. It takes the title and throws a recognition function into the Recognize button. Took away into /utils directory

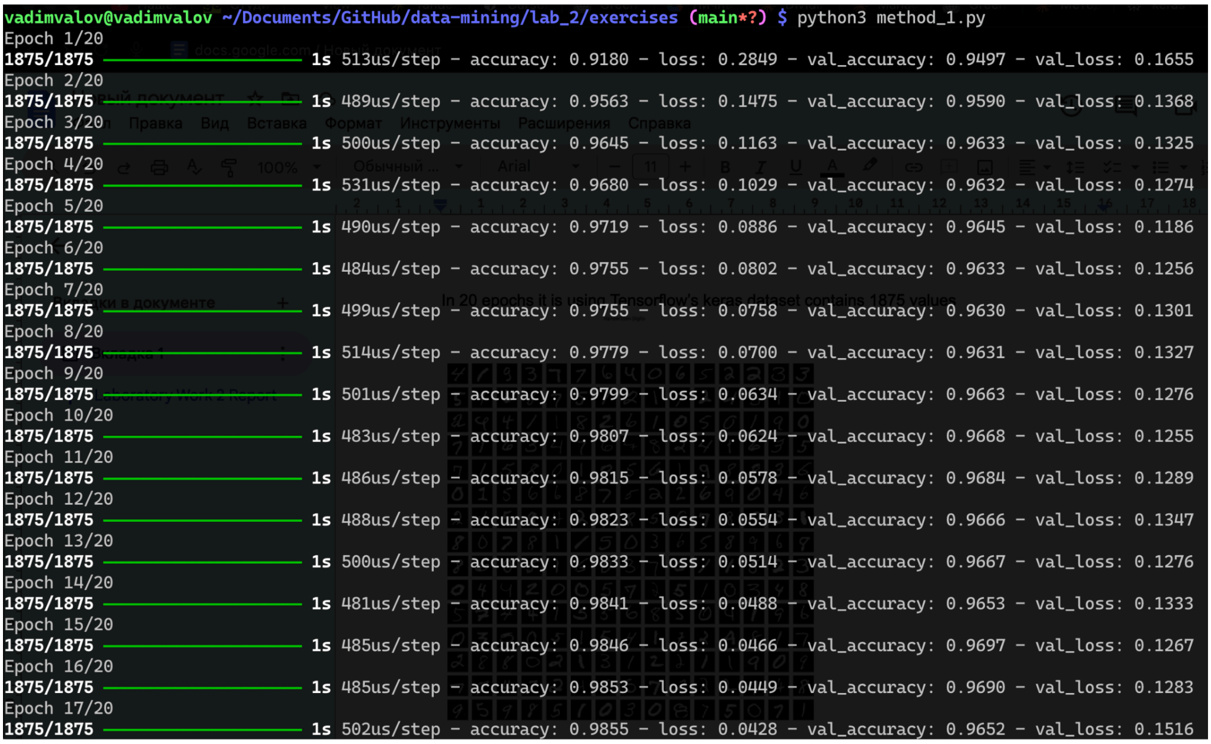


Method 1 (/lab_2/exercises/method_1.py)

Then implemented method 1 using ANN (/lab_2/exercises/method_1.py)
In 20 epochs it is using Tensorflow’s keras dataset contains 1875 values
Input: 784 neurons (28×28 flattened)
Hidden layer 1: 128 neurons
Hidden layer 2: 40 neurons (ReLU activation)
Output: 10 neurons (softmax)

Accuracy: ~98.5% (not less than 97.8% after 10 tests)

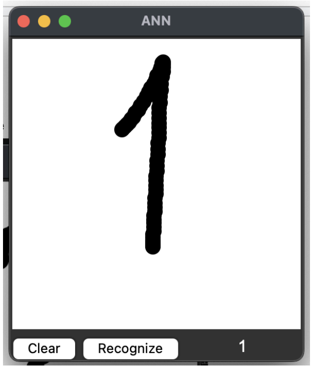


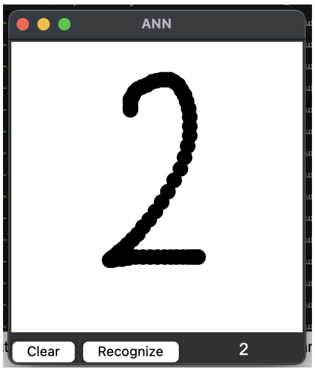


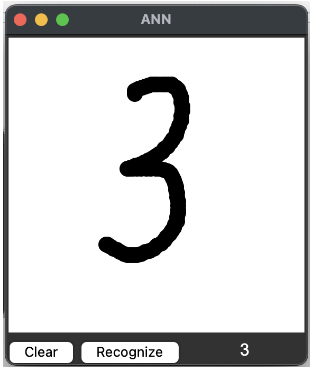
```
vadimvalov@vadimvalov ~/Documents/GitHub/data-mining/lab_2/exercises (main*?) $ python3 method_1.py
Epoch 1/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 513us/step - accuracy: 0.9180 - loss: 0.2849 - val_accuracy: 0.9497 - val_loss: 0.1655
Epoch 2/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 489us/step - accuracy: 0.9563 - loss: 0.1475 - val_accuracy: 0.9590 - val_loss: 0.1368
Epoch 3/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 500us/step - accuracy: 0.9645 - loss: 0.1163 - val_accuracy: 0.9633 - val_loss: 0.1325
Epoch 4/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 531us/step - accuracy: 0.9680 - loss: 0.1029 - val_accuracy: 0.9632 - val_loss: 0.1274
Epoch 5/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 490us/step - accuracy: 0.9719 - loss: 0.0886 - val_accuracy: 0.9645 - val_loss: 0.1186
Epoch 6/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 484us/step - accuracy: 0.9755 - loss: 0.0802 - val_accuracy: 0.9633 - val_loss: 0.1256
Epoch 7/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 499us/step - accuracy: 0.9755 - loss: 0.0758 - val_accuracy: 0.9630 - val_loss: 0.1301
Epoch 8/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 514us/step - accuracy: 0.9779 - loss: 0.0700 - val_accuracy: 0.9631 - val_loss: 0.1327
Epoch 9/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 501us/step - accuracy: 0.9799 - loss: 0.0634 - val_accuracy: 0.9663 - val_loss: 0.1276
Epoch 10/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 483us/step - accuracy: 0.9807 - loss: 0.0624 - val_accuracy: 0.9668 - val_loss: 0.1255
Epoch 11/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 486us/step - accuracy: 0.9815 - loss: 0.0578 - val_accuracy: 0.9684 - val_loss: 0.1289
Epoch 12/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 488us/step - accuracy: 0.9823 - loss: 0.0554 - val_accuracy: 0.9666 - val_loss: 0.1347
Epoch 13/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 500us/step - accuracy: 0.9833 - loss: 0.0514 - val_accuracy: 0.9667 - val_loss: 0.1276
Epoch 14/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 481us/step - accuracy: 0.9841 - loss: 0.0488 - val_accuracy: 0.9653 - val_loss: 0.1333
Epoch 15/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 485us/step - accuracy: 0.9846 - loss: 0.0466 - val_accuracy: 0.9697 - val_loss: 0.1267
Epoch 16/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 485us/step - accuracy: 0.9853 - loss: 0.0449 - val_accuracy: 0.9690 - val_loss: 0.1283
Epoch 17/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 502us/step - accuracy: 0.9855 - loss: 0.0428 - val_accuracy: 0.9652 - val_loss: 0.1516
```

Creates the learned data, stored both in _pycache. Created mnist_ann.keras file with dataset itself.

Then it does recognize the image with ~98% accuracy:



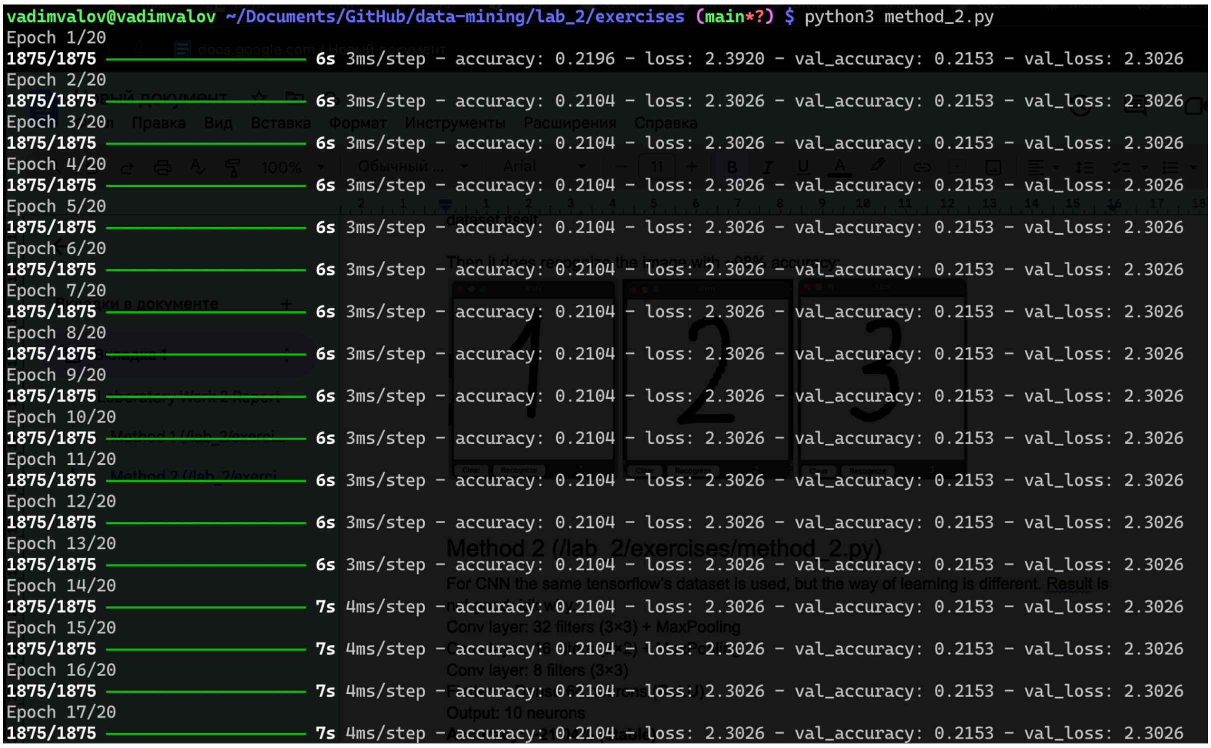




Method 2 (/lab_2/exercises/method_2.py)

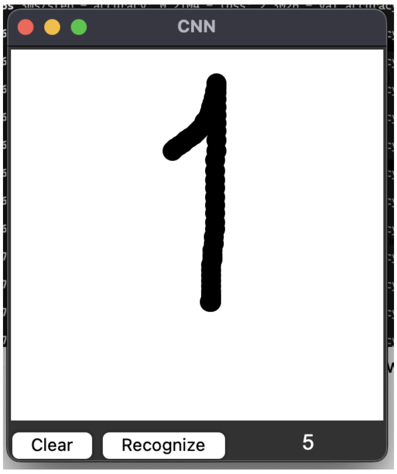
Before starting, the pycache must be cleared.

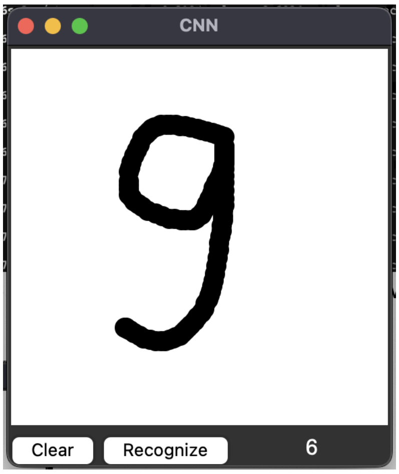
For CNN the same tensorflow's dataset is used, but the way of learning is different. Result is not good, idk why.
Conv layer: 32 filters (3×3) + MaxPooling
Conv layer: 16 filters (2×2) + MaxPooling
Conv layer: 8 filters (3×3)
Flatten + Dense 64 neurons (ReLU)
Output: 10 neurons
Accuracy: ~21.04% (stable)

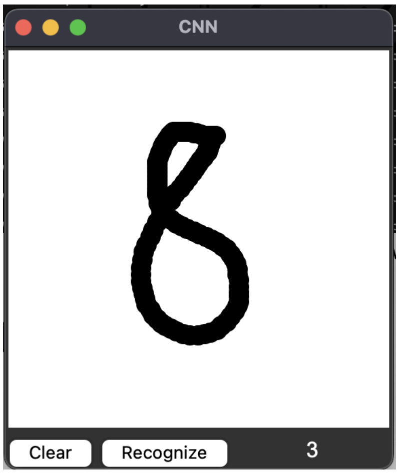


```
vadimvalov@vadimvalov ~/Documents/GitHub/data-mining/lab_2/exercises (main*?) $ python3 method_2.py
Epoch 1/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.2196 - loss: 2.3920 - val_accuracy: 0.2153 - val_loss: 2.3026
Epoch 2/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.2104 - loss: 2.3026 - val_accuracy: 0.2153 - val_loss: 2.3026
Epoch 3/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.2104 - loss: 2.3026 - val_accuracy: 0.2153 - val_loss: 2.3026
Epoch 4/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.2104 - loss: 2.3026 - val_accuracy: 0.2153 - val_loss: 2.3026
Epoch 5/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.2104 - loss: 2.3026 - val_accuracy: 0.2153 - val_loss: 2.3026
Epoch 6/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.2104 - loss: 2.3026 - val_accuracy: 0.2153 - val_loss: 2.3026
Epoch 7/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.2104 - loss: 2.3026 - val_accuracy: 0.2153 - val_loss: 2.3026
Epoch 8/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.2104 - loss: 2.3026 - val_accuracy: 0.2153 - val_loss: 2.3026
Epoch 9/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.2104 - loss: 2.3026 - val_accuracy: 0.2153 - val_loss: 2.3026
Epoch 10/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.2104 - loss: 2.3026 - val_accuracy: 0.2153 - val_loss: 2.3026
Epoch 11/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.2104 - loss: 2.3026 - val_accuracy: 0.2153 - val_loss: 2.3026
Epoch 12/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.2104 - loss: 2.3026 - val_accuracy: 0.2153 - val_loss: 2.3026
Epoch 13/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.2104 - loss: 2.3026 - val_accuracy: 0.2153 - val_loss: 2.3026
Epoch 14/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.2104 - loss: 2.3026 - val_accuracy: 0.2153 - val_loss: 2.3026
Epoch 15/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.2104 - loss: 2.3026 - val_accuracy: 0.2153 - val_loss: 2.3026
Epoch 16/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.2104 - loss: 2.3026 - val_accuracy: 0.2153 - val_loss: 2.3026
Epoch 17/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.2104 - loss: 2.3026 - val_accuracy: 0.2153 - val_loss: 2.3026
```

As a result, it takes up to 7 times more to learn and the accuracy is lower. Loss is 2.3026 which is bad.







Results

Well idk why the result of CNN is worse than with ANN, most likely my own mistake or misconfiguration or smth. During implementing the methods and practicing, I did implement number recognition twice, so that because Experiment part is empty.