Python Alv🐍🐍ros

DOCUMENTATION

Title and authors

Squad:

- Paweł "Anaconda" Kosoń
- Michał "Boa" Roguz
- Kacper "Rattlesnake" Goraj
- Szymon "Cobra" Pawelec

Project description

The main goal was to create a responsive Android application with the help of Machine Learning algorithms and neural nets. Our task was to recognise letters and digits from images with accuration ratio. We've merged some potentially different aspects together such as having fun and learning Python.

Initial assumptions & implementation

For the project's purposes we've chosen some useful environments such as Colab, replit and Visual Studio. We've decided to use convenient libraries (Keras, Kivy, Tensorflow- for image operating and NumPy, CV2, Matplotlib- for visualizing data) so as to approach the problem properly and get satisfying results. What's more, we've trained a neural net on NIST dataset. This workflow let us achieve some of assumpted goals.

Project analysis

The input data in our application are scans containing handwritten letters or digits. You will be able to load it using the "Load scan" button.The output data in our application will be the recognized character and the precision of solution. To start the recognition, press the "Recognise" button. In our program we saved images to variables using the "cv2" library and its tools.The user interface is easy to use. Button for reading, recognizing and cleaning the canvas with the image. Additionally, we have two windows showing gained result and recognition accuracy.

We divided the project into smaller parts, which we consistently implemented in the following weeks. Illustrative breakdown of the project implementation:
- existing database selection and import
- uppercase letters training
- lowercase letters training
- digit training
- training everything at once
- creating a GUI
- a jointly trained model with a graphical interface

How to run the app?
At the command line, launch the gui.py file.

WORK DISTRIBUTION:

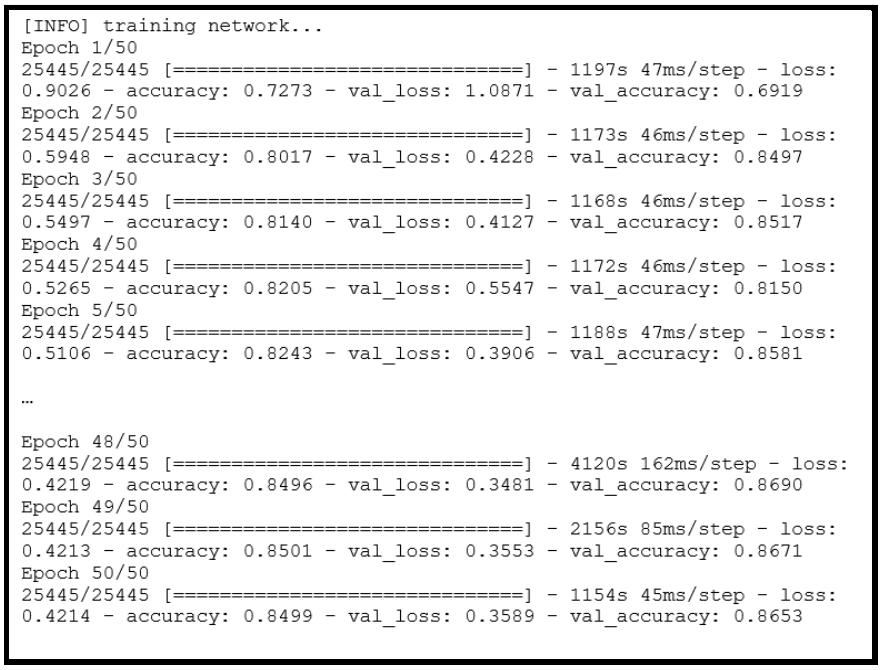
Paweł - ML code implementation, research of algorithms and methodology, training neural network;
Michał - Editor-in-chief of the documentation, ML code implementation, selection of libraries, training neural net;
Kacper - processing the data, analyzing handwritten signs, preparation of documentation, testing of solution;
Szymon - Chief Design Officer, creating application interface, combining model & GUI together, implementing necessary algorithms for proper way of processing images;

Correct Roadmap

Date	18.11.2021-26.11.2021	27.11.2021-06.12.2021	07.12.2021-23.12.2021	24.12.2021-03.01.2022	04.01.2022-11.01.2022
Name	Grass snake stage	Viper phase	Coral Snake's Time	The Taipan period	Moment of python's
Goal	Choosing project environment	Learning libraries	First sketch of the project and data preparation	Building up a working application	Testing, code improvement & last corrections
Metrics	Configure the environment	Understanding requirements of the project	Activations	Activations	Downloads and launch

Development

Gradually, after selecting and understanding a dataset with numbers and uppercase and lowercase letters, we gradually started training neural networks in Python. We trained the datasets separately at the beginning and all together at the end.



```
[INFO] training network...
Epoch 1/50
25445/25445 [==============================] - 1197s 47ms/step - loss:
0.9026 - accuracy: 0.7273 - val_loss: 1.0871 - val_accuracy: 0.6919
Epoch 2/50
25445/25445 [==============================] - 1173s 46ms/step - loss:
0.5948 - accuracy: 0.8017 - val_loss: 0.4228 - val_accuracy: 0.8497
Epoch 3/50
25445/25445 [==============================] - 1168s 46ms/step - loss:
0.5497 - accuracy: 0.8140 - val_loss: 0.4127 - val_accuracy: 0.8517
Epoch 4/50
25445/25445 [==============================] - 1172s 46ms/step - loss:
0.5265 - accuracy: 0.8205 - val_loss: 0.5547 - val_accuracy: 0.8150
Epoch 5/50
25445/25445 [==============================] - 1188s 47ms/step - loss:
0.5106 - accuracy: 0.8243 - val_loss: 0.3906 - val_accuracy: 0.8581

…

Epoch 48/50
25445/25445 [==============================] - 4120s 162ms/step - loss:
0.4219 - accuracy: 0.8496 - val_loss: 0.3481 - val_accuracy: 0.8690
Epoch 49/50
25445/25445 [==============================] - 2156s 85ms/step - loss:
0.4213 - accuracy: 0.8501 - val_loss: 0.3553 - val_accuracy: 0.8671
Epoch 50/50
25445/25445 [==============================] - 1154s 45ms/step - loss:
0.4214 - accuracy: 0.8499 - val_loss: 0.3589 - val_accuracy: 0.8653
```

Image. Screen with training network

We have created a trained model called "kompletny.model".

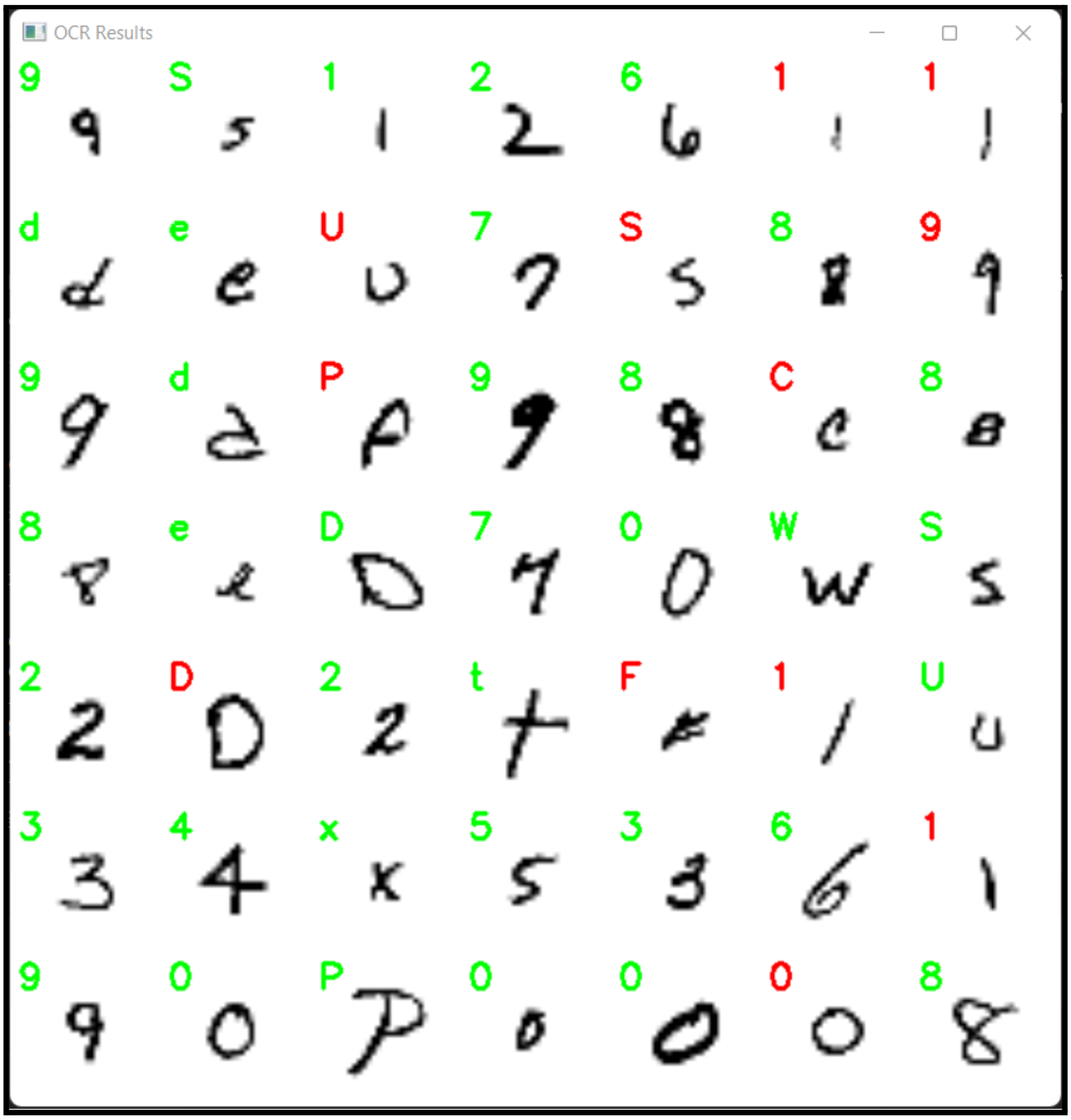


Image. Screen with test recognition

In the next step, we combined this model with the previously created GUI. Later, we expanded the GUI to draw letters and numbers aside from loading only handwritten scans.
We created the functionality of writing your own characters and the ability to recognize words by dividing them into individual characters, but they did not work properly, so we gave them up.

Testing

Parallel to writing the code, we checked if it worked. Later, when testing letters and numbers, we used examples from the dataset or handwritten.

Implementation, report and conclusions

what succeeded:

The application recognizes numbers and letters (basic assumptions completed!) with an acceptable accuracy. All buttons work. Program works very fast.

what failed:

The application doesn't recognize interconnected letters or words. Clients can't draw their own letters. We've tried to add both functionalities, but encountered many problems and decided to abandon this way of thinking. We've focused on fulfilling basic requirements.