Traffic Signs Recognition CNN

Softcomputing
Project 1

Authors:
Weronika Kołodziej 263232
Szymon Pardyka 235376

Date: 06.12.2021

Table of content

1. Description of the task

For the project we have chosen recognition of traffic signs. This is a challenging real-world problem. Traffic signs show a wide range of variations between classes in terms of color, shape, and the presence of pictograms or text. However, the signs can be splitted into a few subsets that have specific features and make every sign being part of the given class similar to its other representatives.

Technologies used to create project:

- Python v3.9.0 - overall usage
- Keras package v2.7.0 - convolutional neural networks
- Pandas package - reading data
- cv package v2 - data preparation

2. Chosen dataset

The chosen dataset is GTSRB dataset wchich stands for German Traffic Sign Recognition Benchmark.

Dataset contains 43 different signs which include 39 209 images in PPM format total, however the amount of images is not evenly distributed among the signs. The image sections contain only the traffic signs.

The size of the traffic signs varies between 15×15 and 222×193 pixels. The images contain 10% margin (at least 5 pixels) around the traffic sign to allow for the usage of edge detectors. The images are not necessarily square.

Dataset contains both high and low quality images with different sizes and clarity.

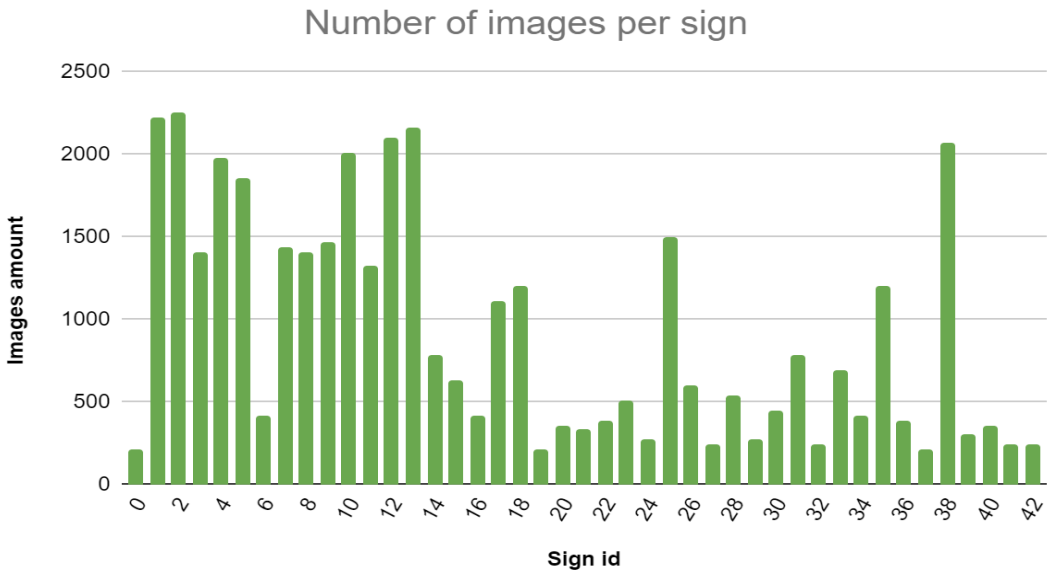

Fig. 1 - Number of images for each sign present in the dataset.

3. Samples of the data from dataset

The main split separates the data into the full training set and the test set. The training set is ordered by class.

In figure 1, we can notice that there is a significant imbalance across classes in the training set. Some classes have less than 200 images, while others have over 2000. This means that our model could be biased towards over-represented classes, especially when it is unsure in its predictions as we can notice later in the result.

Below (in the figure 2) we present some of the pictures from the training set. Some pictures in the dataset are quite dark, some of them on the contrary.

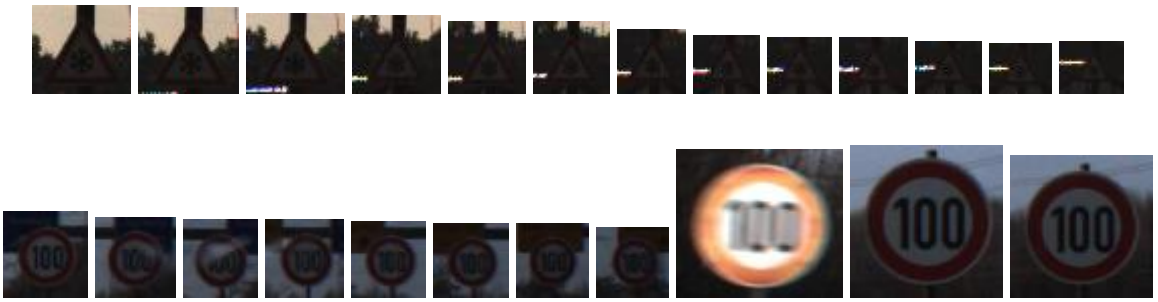

Fig. 2 - Part of the training dataset.

4. What has been done

The large dataset was selected. In the project two Python scripts were created using mainly the Tensorflow package. First script is named “main.py” and is responsible for loading, preparing and splitting the dataset and finally teaching the model. After model training, script uses the data that was earlier separated for the testing purposes, and runs model prediction. At the end the model is evaluated and saved to a separate file.

Second script is used to prepare and recognise a given image. At first the image is being processed to be readable for the model. nextly model is being loaded from the file. Finally script runs the predict function and responds with the following data:

- Table of 43 values - each entry represents a fractional value which states in how much percent the given image fits the specified under given index sign pattern.
- Highest value present in the result array - value between 0 - 1. The higher it is, the more certain the model is about image presenting traffic sign under the values index.
- Index value - value between 0 and 42. Each of these represents a different sign from the dataset.

In both cases the image/s have to be pre-processed for the neural network. All of the images must be in the same amount of dimensions. Dataset consists of images with dimensions from 16×16×3 to 128×128×3 which requires adjusting all of them to the same dimension. 64×64×3 - it is a middle between the border values which means that the images will not lose too much of the data or will not need a lot of stretching, which makes this dimension the best choice.

5. Teaching the Model

Model is being formed and trained in the “main.py” file. The configuration is as follows:

- Epochs: 10
- Batch size: 16
- Pool size: 2×2
- Hidden units number: 2048, 1024, 128
- Output units number: 43

Model works on randomly separated data. Images are divided into three subsets - learning, validation and testing in 0.6×0.2×0.2 proportions. Each epoch is learning on randomly selected 60% of images, and validating the results on the validation set. Process is being repeated ten times, in which every new generation comes with better results - higher accuracy and smaller losses.

Model:

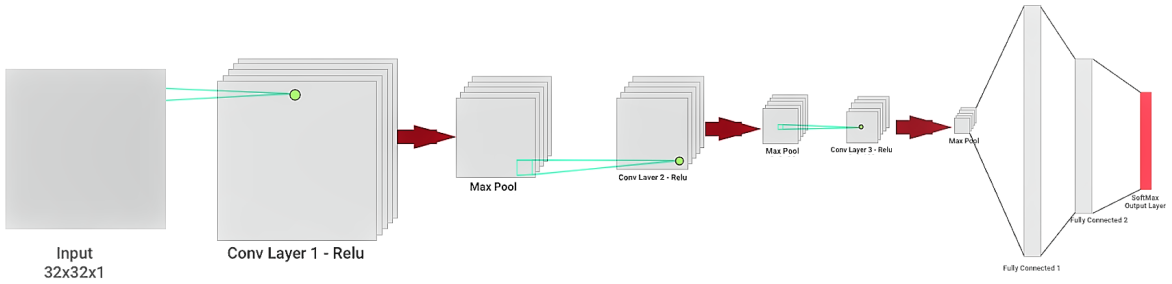

Fig. 3 - Training model.

Training output looks as follows:

```
Epoch 1/10
1471/1471 [==============================] - 206s 139ms/step - loss: 2.6046 - accuracy: 0.2970 - val_loss: 1.2679 - val_accuracy: 0.6343
Epoch 2/10
1471/1471 [==============================] - 204s 139ms/step - loss: 1.1059 - accuracy: 0.6601 - val_loss: 0.4345 - val_accuracy: 0.8670
Epoch 3/10
1471/1471 [==============================] - 205s 139ms/step - loss: 0.4397 - accuracy: 0.8632 - val_loss: 0.1349 - val_accuracy: 0.9594
Epoch 4/10
1471/1471 [==============================] - 204s 139ms/step - loss: 0.2201 - accuracy: 0.9302 - val_loss: 0.0783 - val_accuracy: 0.9779
Epoch 5/10
1471/1471 [==============================] - 203s 138ms/step - loss: 0.1450 - accuracy: 0.9556 - val_loss: 0.0591 - val_accuracy: 0.9842
Epoch 6/10
1471/1471 [==============================] - 205s 140ms/step - loss: 0.1004 - accuracy: 0.9687 - val_loss: 0.0469 - val_accuracy: 0.9860
Epoch 7/10
1471/1471 [==============================] - 204s 139ms/step - loss: 0.0732 - accuracy: 0.9772 - val_loss: 0.0420 - val_accuracy: 0.9894
Epoch 8/10
1471/1471 [==============================] - 205s 139ms/step - loss: 0.0573 - accuracy: 0.9820 - val_loss: 0.0378 - val_accuracy: 0.9903
Epoch 9/10
1471/1471 [==============================] - 203s 138ms/step - loss: 0.0499 - accuracy: 0.9842 - val_loss: 0.0472 - val_accuracy: 0.9883
Epoch 10/10
1471/1471 [==============================] - 205s 139ms/step - loss: 0.0374 - accuracy: 0.9880 - val_loss: 0.0305 - val_accuracy: 0.9926
```

Fig. 4 - Model training output.

At the end the model is being evaluated. Evaluation results are:

- loss: *0.0411*
- accuracy: *0.9911*

```
246/246 [==============================] - 11s 43ms/step - loss: 0.0411 - accuracy: 0.9911
```

Fig. 5 - Model evaluation output.

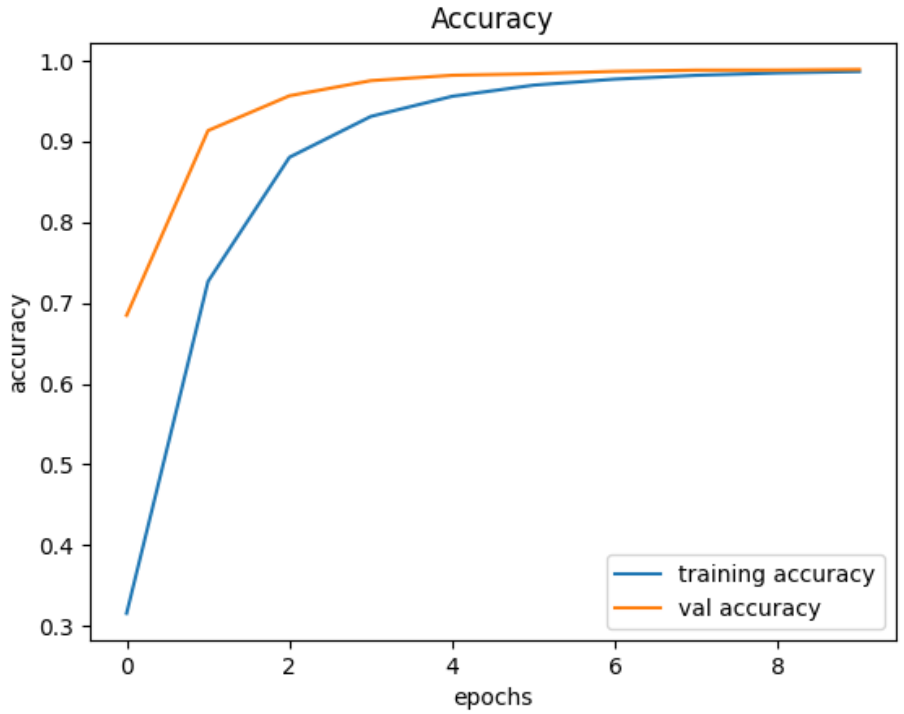

Fig. 6 - Training accuracy.

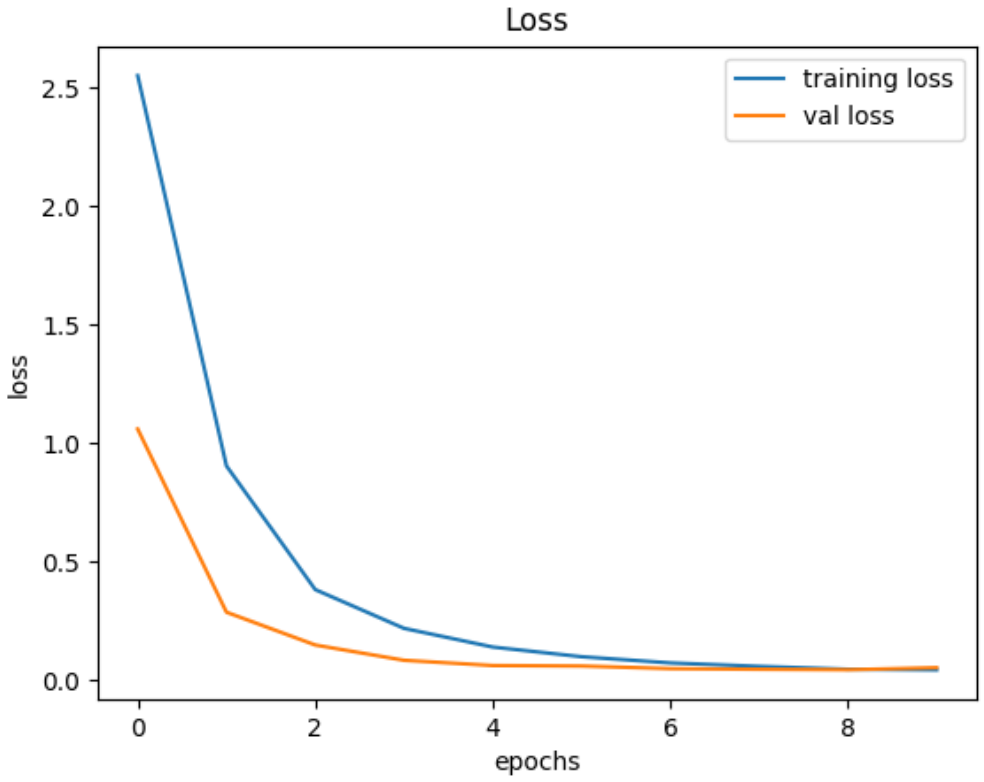

Fig. 7 - Training loss.

6. Testing the Model - Result

After testing the model with data present in dataset, the result present themselves as follows:

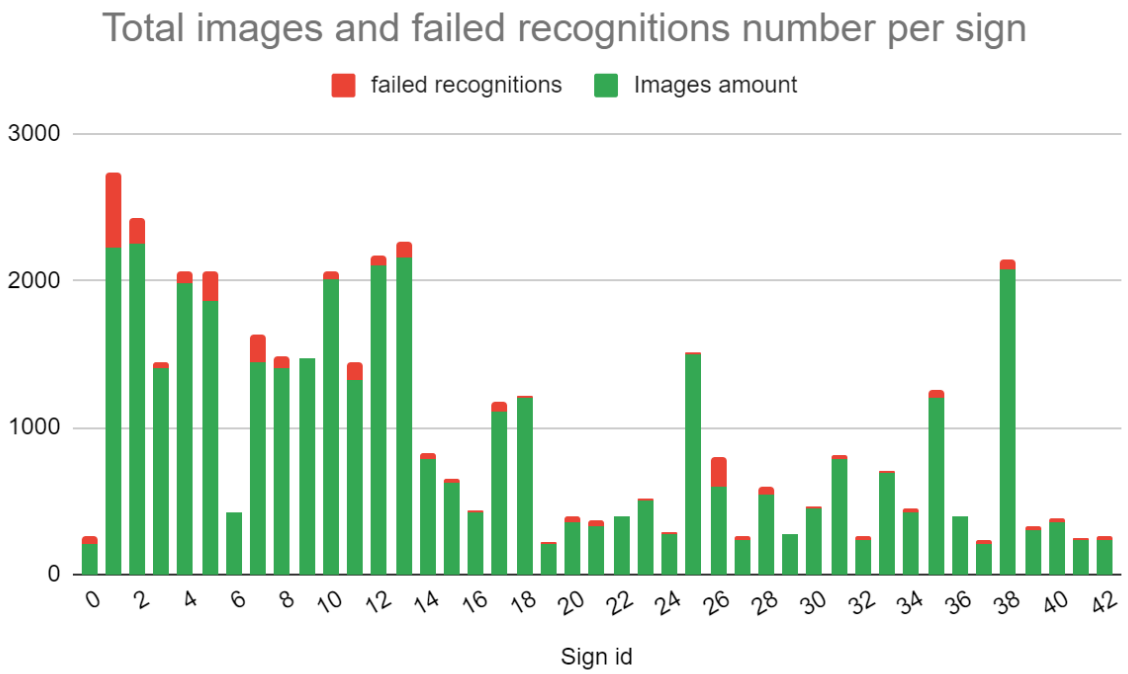

Fig.8 - Model testing results.

Chart represents how many images of a given sign were recognised wrongly. On average, the model was accurate in 99.1% of all cases. There are a few signs that were problematic for the model to recognise them, however they are low in number (signs with 1,2,7, 26 IDs) and seem to be independent to the images amount which suggests that part of the mentioned signs images were in low quality.

Test with randomly traffic sign: **Recognition success**

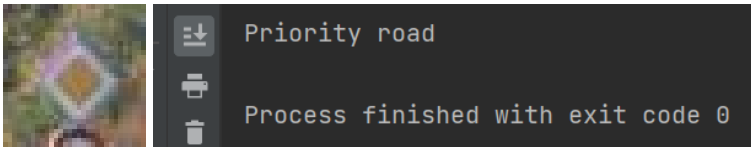

Fig.9 - Tested sign and prediction.

To check how well the model works, standard images have been replaced with rotated ones. To proceed with the tests, signs with both very small and very big

amounts of images have been selected. Additionally few signs with the best and the worst recognition results.

Sign 0: 211 total images, 50 failed recognitions

- 5 degrees rotation - 89 mistakes
- 10 degrees rotation - 152 mistakes
- 15 degrees rotation - 180 mistakes

Sign 1: 2221 total images, 514 failed recognitions

- 5 degrees rotation - 587 mistakes
- 10 degrees rotation - 791 mistakes
- 15 degrees rotation - 1364 mistakes

Sign 6: 421 total images, 0 failed recognitions

- 5 degrees rotation - 3 mistakes
- 10 degrees rotation - 7 mistakes
- 15 degrees rotation - 33 mistakes

Sign 25: 1501 total images, 16 failed recognitions

- 5 degrees rotation - 46 mistakes
- 10 degrees rotation - 69 mistakes
- 15 degrees rotation -162 mistakes

Sign 38: 2071 total images, 79 failed recognitions

- 5 degrees rotation - 1766 mistakes
- 10 degrees rotation - 1802 mistakes
- 15 degrees rotation - 1805 mistakes

Sign 38: 2071 total images, 79 failed recognitions

- 5 degrees rotation - 1766 mistakes
- 10 degrees rotation - 1802 mistakes
- 15 degrees rotation - 1805 mistakes

Tested darkened sign: **Recognition success**

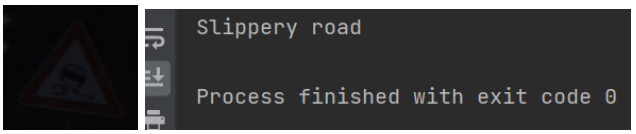

Fig.10 - Tested sign and prediction.

Tested polish sign from official site http://znaki-drogowe.org/:**Recognition failed**

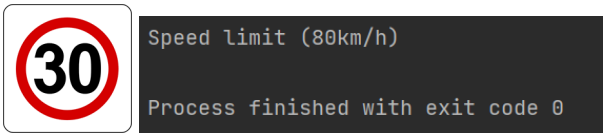

Fig.11- Tested sign and prediction.

Brightened sign: **Recognition Success**

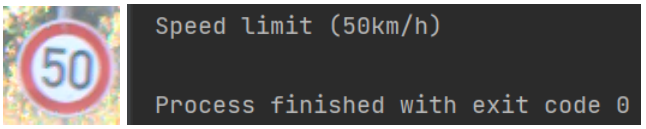

Fig.12 - Tested sign and prediction.

Rotated sign - 90 degrees: **Recognition Success**

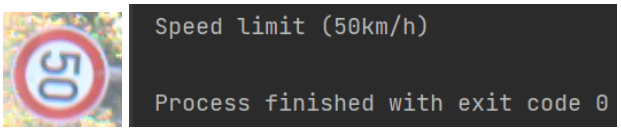

Fig.13 - Tested sign and prediction.

Distorted sign: **Recognition Success**

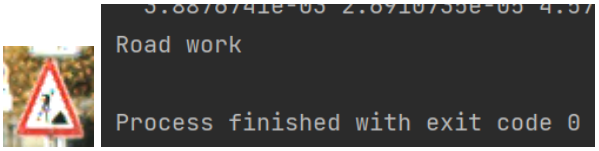

Fig.14 - Tested sign and prediction.

7. Summary

Taking into account the above considerations, we prepared a wide dataset of traffic signs. The results are quite satisfying, although there are some cases wrongly recognized. We suppose it is caused by the low quality of some of the pictures or uneven image number distribution among signs. It may cause the model to be better at recognizing some signs more than others. Rotating the signs can also make recognition of the sign a much harder task. The more the image has been rotated, the harder it was for the model to recognise it properly. However, it is necessary to

mention that some of the signs when rotated with specific angle look exactly like others, for example pass on right only (number 38 in tests).

This confirms convolutional neural networks are able to learn task-specific features from raw data with quite pleasing results. To obtain more precise results we could improve pictures' quality and/or increase data for some sign classes dedicated for training.