CONTENT

PROJECT GOALS

The main goal was to create a tool for recognizing Polish road signs, using a custom variation of the Convolutional Neural Network model (as it will turn out later, not only)

The final result should be a prediction of one or more traffic signs found in the selected image and output them for the user

Eventually, it was expected to do a test on real car camera images and verify the results

APPROACH

BALANCED DATASET

The project was launched with a search for a balanced dataset. However, I really wanted them to be Polish road signs, because I wanted to then finally test the model on our car images.

SIGN DETECTION

I determined that sign recognition would be much simpler with the sign itself in the picture (this would lead to high efficiency of the model and its ease in catching more details). For this reason, I decided to use the model for detection before classifying the signs

SIGN CLASSIFICATION

There were many options to choose from when it came to a model for this task. I decided on the Spline Neural Network model because it is popular for its accuracy for classification and there is a lot of content about it that I can learn from

ENCOUNTERED PROBLEMS

Throughout the process, I encountered many problems that I would like to talk a little about

UNBALANCED DATASET

I have reviewed 4 different dataset options:

- original
- with deleted categories under 100 images
- with augmented all categories to 300 images
- with removal and augmentation both

The learning curves and accuracy of the basic model led me to decision of choosing option 4 with following steps:

I couldn't find a better dataset with Polish road signs - I had to deal with the problem of unbalanced data.

FLOATING NUMBER OF IMAGES PER CLASS

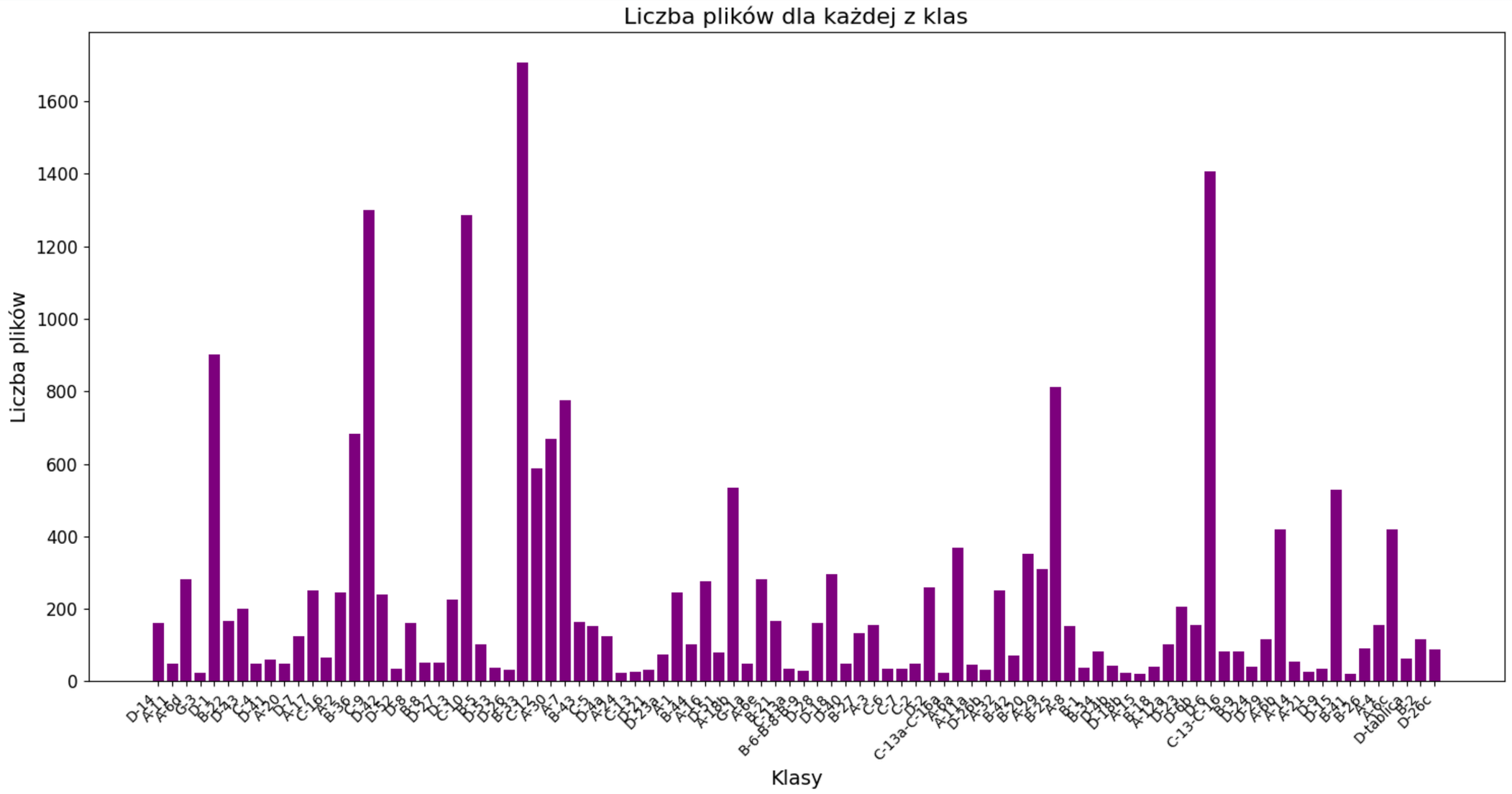
I have decided to remove classes that contain less than 100 images. There was no point in augmenting them, because the model could overfeed on photos that are too similar to each other

REMOVAL

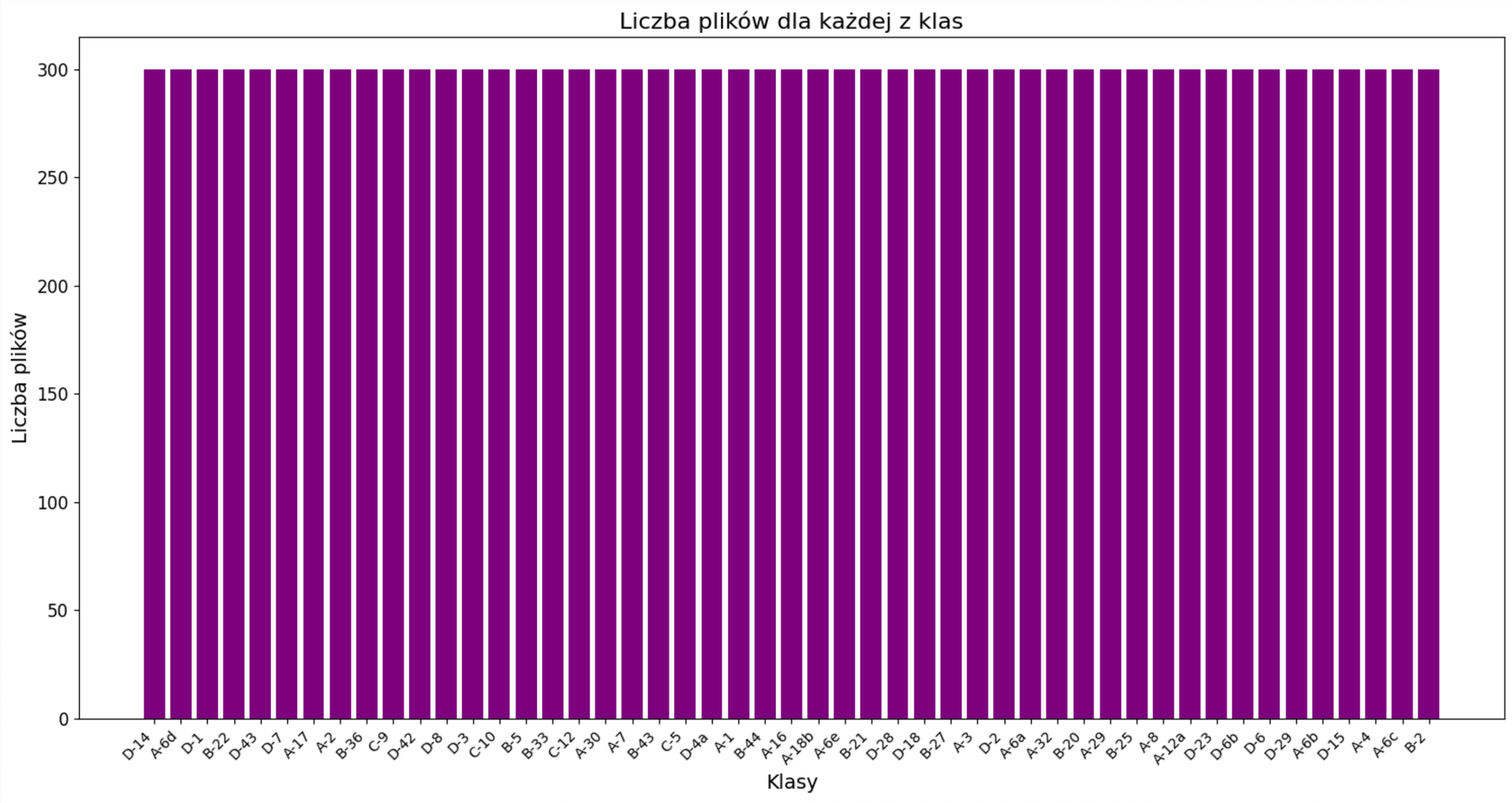
In the remaining classes, I applied a truncation of too many photos to 300, and in classes where photos were missing, I performed augmentation. Operations of gentle rotation, color changes, blurring, shifting and brightness changes helped produce photos up to the designated number of 300 per class

AUGMENTATION

FLOATING NUMBER OF IMAGES PER CLASS



AFTER REMOVAL & AUGMENTATION



This leaves me with 49 from 92 classes of recognized character classes, with 300 images per class

EXAMPLE OF DATA SAMPLES PER CLASS

SHAPING MAIN CNN MODEL

I created my own model based on knowledge from the lectures and any material from the Internet. During the creation process, I did a lot of testing, while building an intuition that I did not have before

At first I built CNN model with 2 convolutional layers, compleck with 2 pooling layers and a fully connected layer at the end

BASIC CNN MODEL

I added additional weave layers, adjusted the number of filters as well as the number of neurons in the last fully connected layer

EXTENDED CNN MODEL

My model was overfiting. I applied L2 regularization, dropout after each convultional layer and in the fully connected layer. I also adjusted the learning rate and number of epochs

FINAL CNN MODEL

BASIC CNN MODEL

```
model1 = Sequential([
    Conv2D(60, (5, 5), input_shape=(32, 32, 3), activation='relu'),
    Conv2D(60, (5, 5), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(30, (3, 3), activation='relu'),
    Conv2D(30, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    Flatten(),
    Dense(500, activation='relu'),
    Dropout(0.5),
    Dense(92, activation='softmax')
])
```

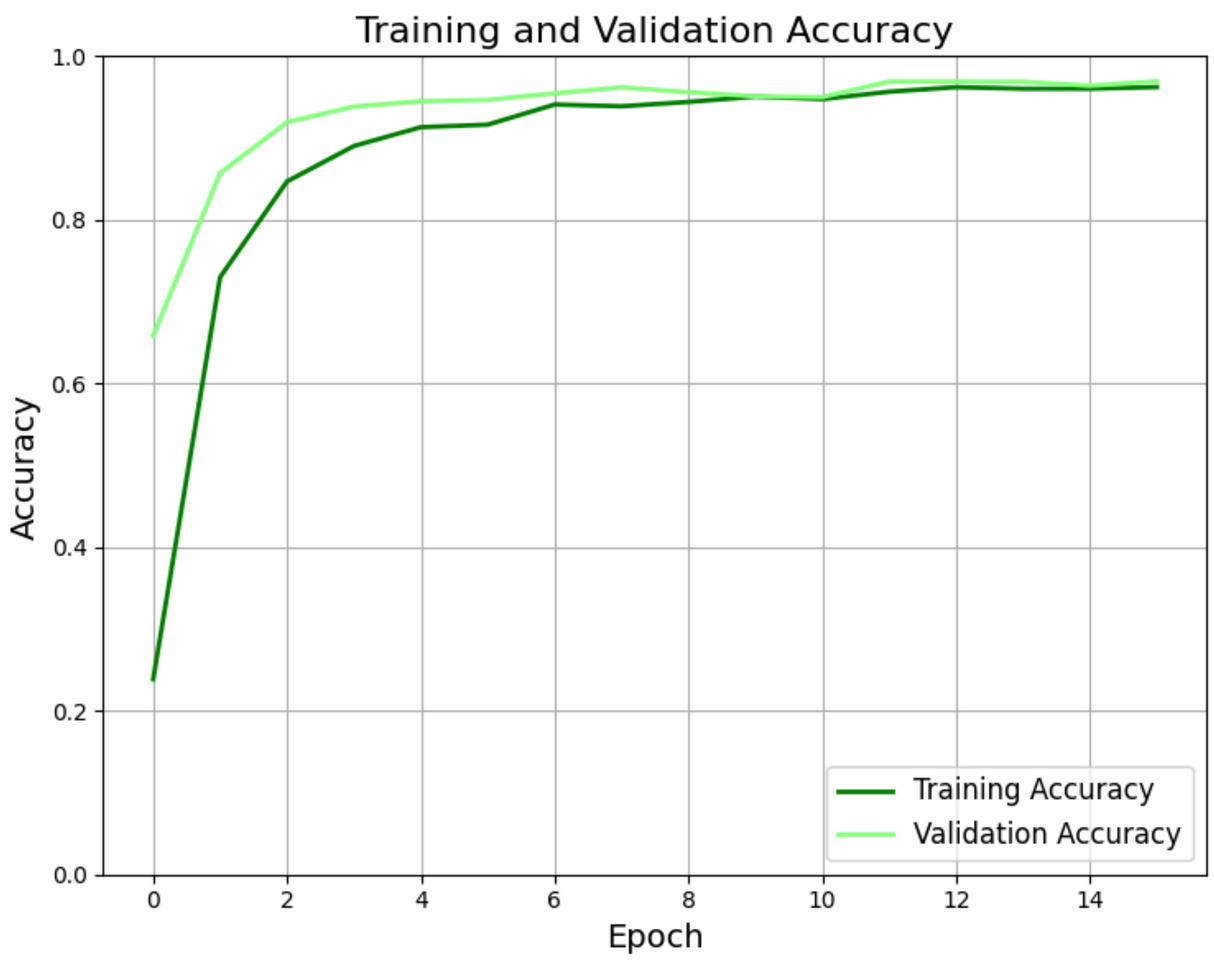
FINAL CNN MODEL

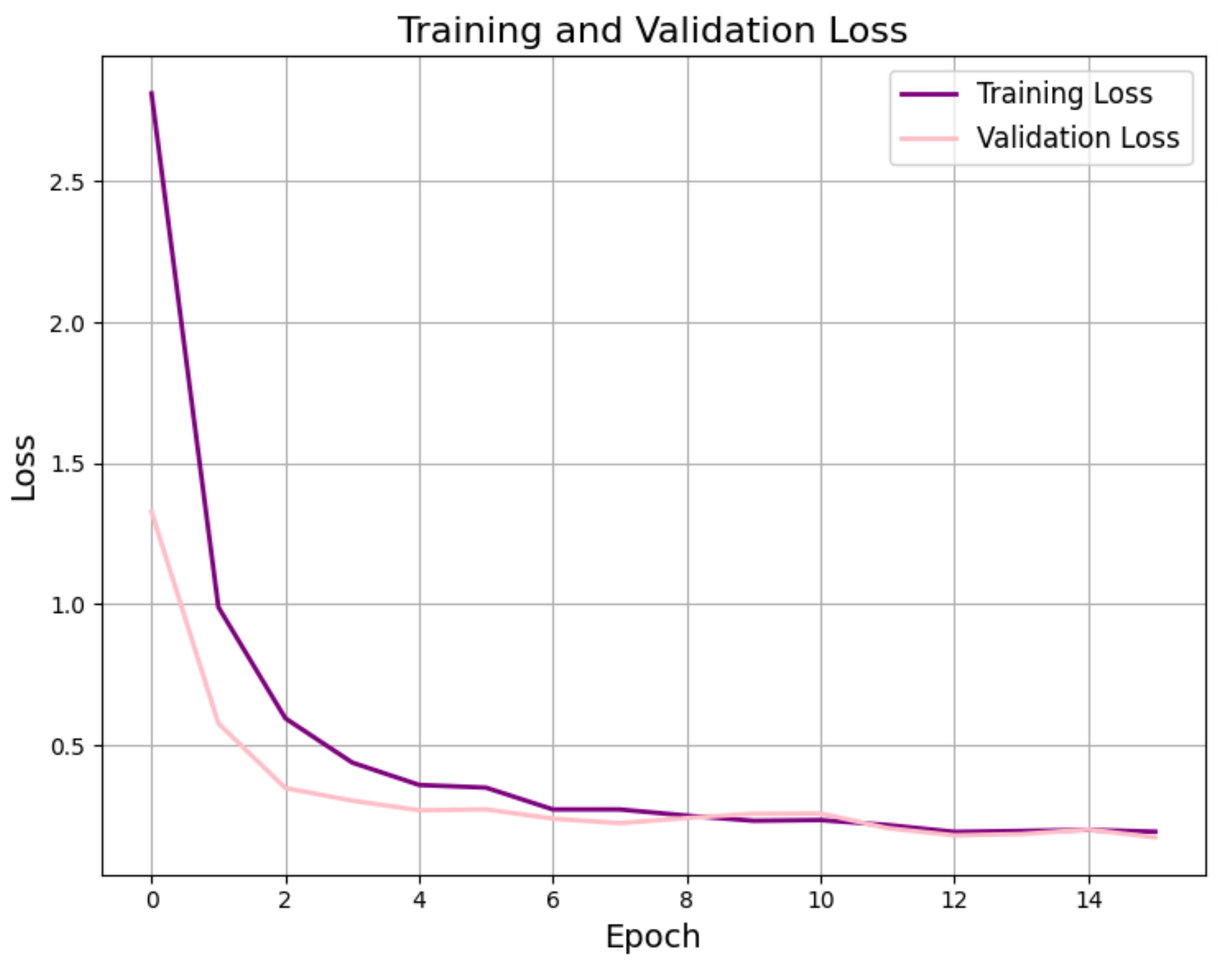
```
model = Sequential([
    Conv2D(32, (5, 5), input_shape=(32, 32, 3), activation='relu', kernel_regularizer=l2(0.001)),
    Conv2D(32, (5, 5), activation='relu', kernel_regularizer=l2(0.001)),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.2),

    Conv2D(64, (3, 3), activation='relu', kernel_regularizer=l2(0.001)),
    Conv2D(64, (3, 3), activation='relu', kernel_regularizer=l2(0.001)),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.2),

    Flatten(),
    Dense(294, activation='relu'),
    Dropout(0.2),
    Dense(49, activation='softmax')
])
```

HYBRID MODEL SUMMARY AND STATISTICS

95%

ACCURACY AND LOSS CURVES OF CNN MODEL





STATISTICS

Ultimately, my hybrid model recognizes 49 classes. Learning the CNN model was made on 8820 photos, and the validation and test sets have each 2940 photos

DATA

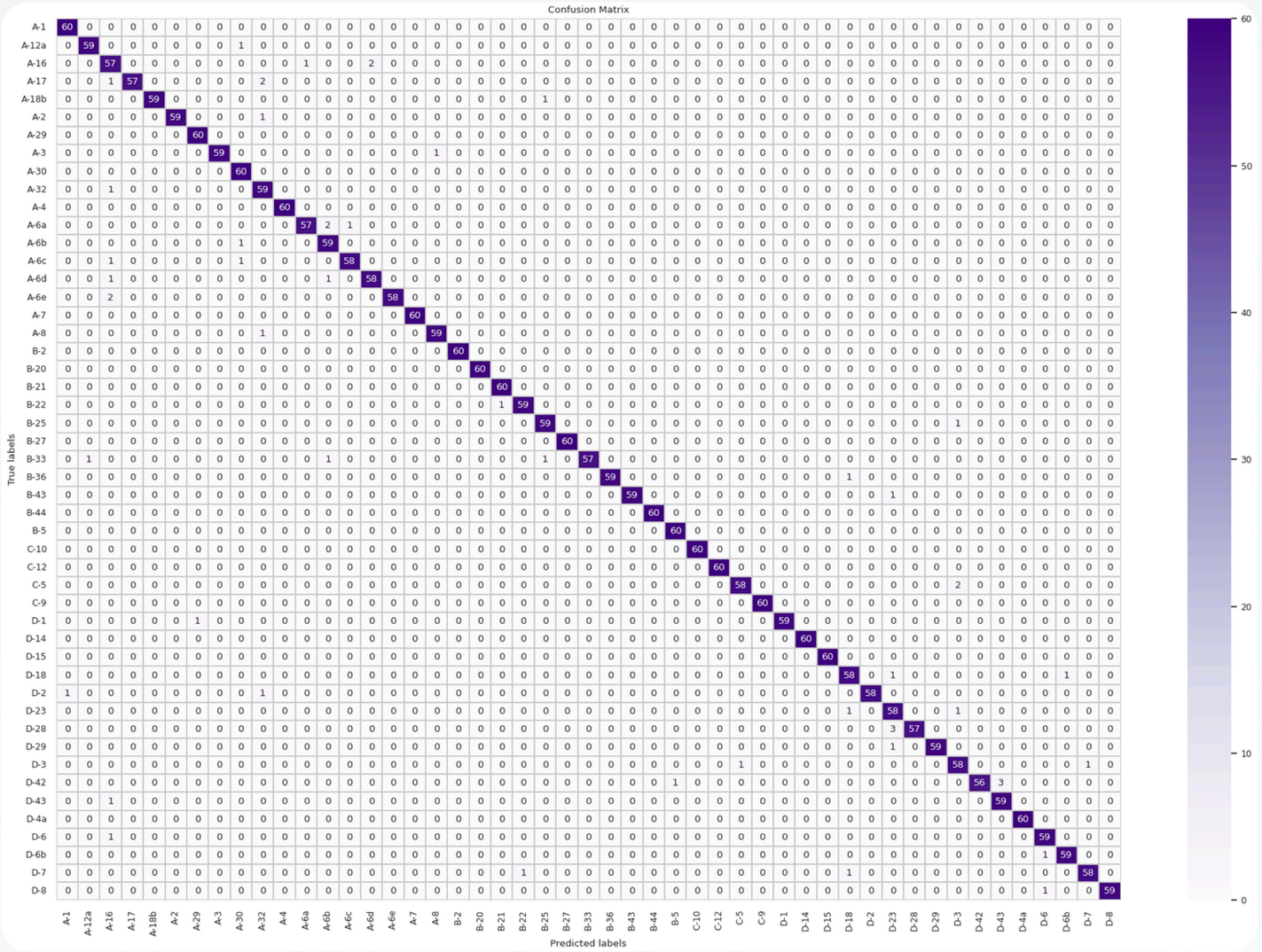
On average, the CNN model was able to achieve about 95% success in recognizing sign classes from cropped images

CNN MODEL ACCURACY

168 layers, 3,005,843 parameters, learning time ~ 20mins. Satisfactory efficiency to detect 81% of the road signs in the photos

YOLOV8 STATS

CONFUSION MATRIX OF CNN MODEL (ON TEST SET)



EXAMPLE PREDICTION WITH HYBRID MODEL

C-12
Ruch okrężny

A-7
Ustąp pierwszeństwa

D-6
Przejście dla pieszych

B-36
Zakaz
zatrzymywania się

IN-CODE PROOF

```
import gc
gc.collect()
torch.cuda.empty_cache()

predict_with_hybrid_model(random_jpg_file_path)
```

DETECTION & PREDICTION

```
--------------------------------------------------
Model predicted: C-12 | Class name: Ruch okrężny
--------------------------------------------------
```

DETECTION & PREDICTION

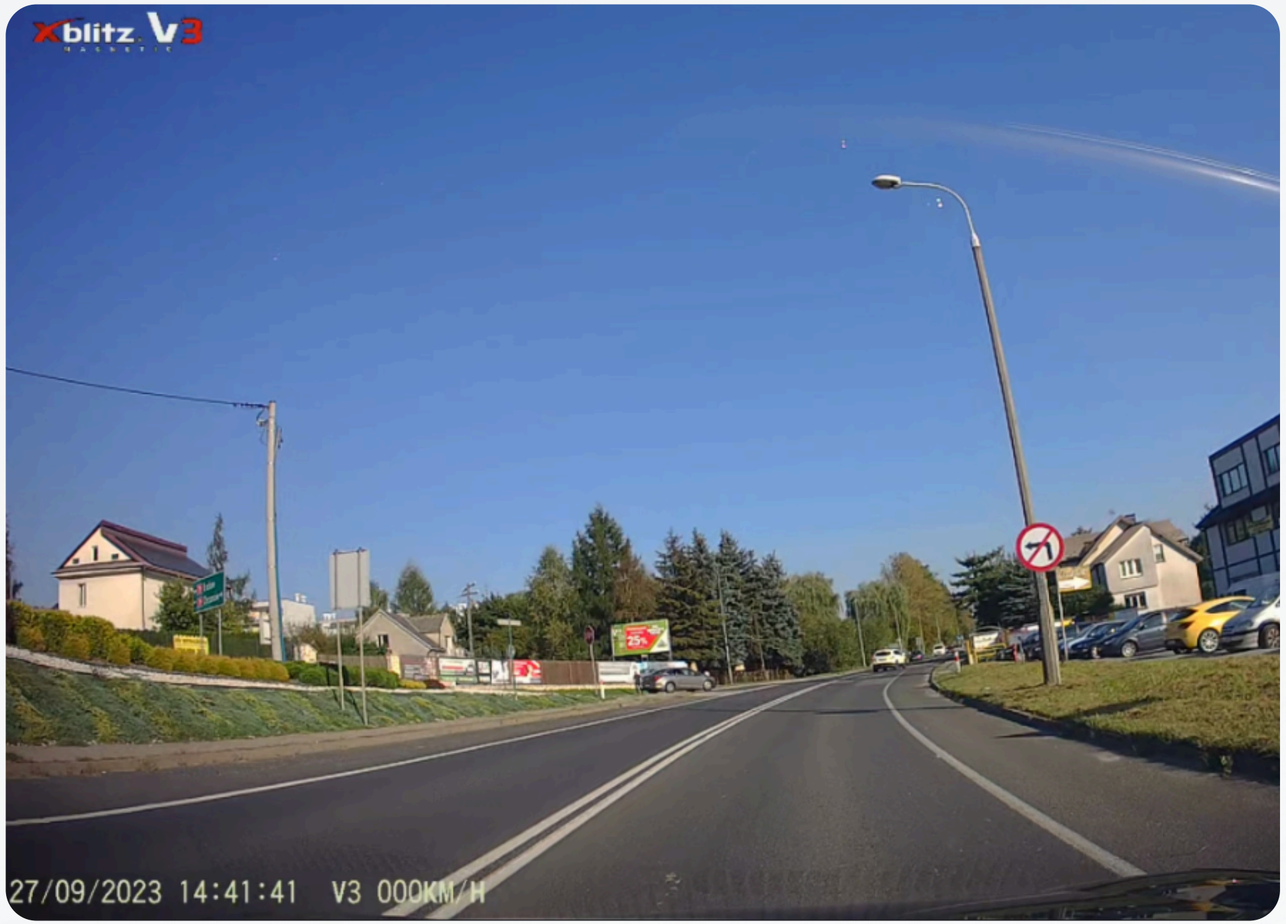
```
------------------------------------------------------
Model predicted: A-7 | Class name: Ustąp pierwszeństwa
------------------------------------------------------
```

DETECTION & PREDICTION

DETECTION & PREDICTION

```
------------------------------------------------------------
Model predicted: B-36 | Class name: Zakaz zatrzymywania się
------------------------------------------------------------
```

2ND PREDICTION EXAMPLE



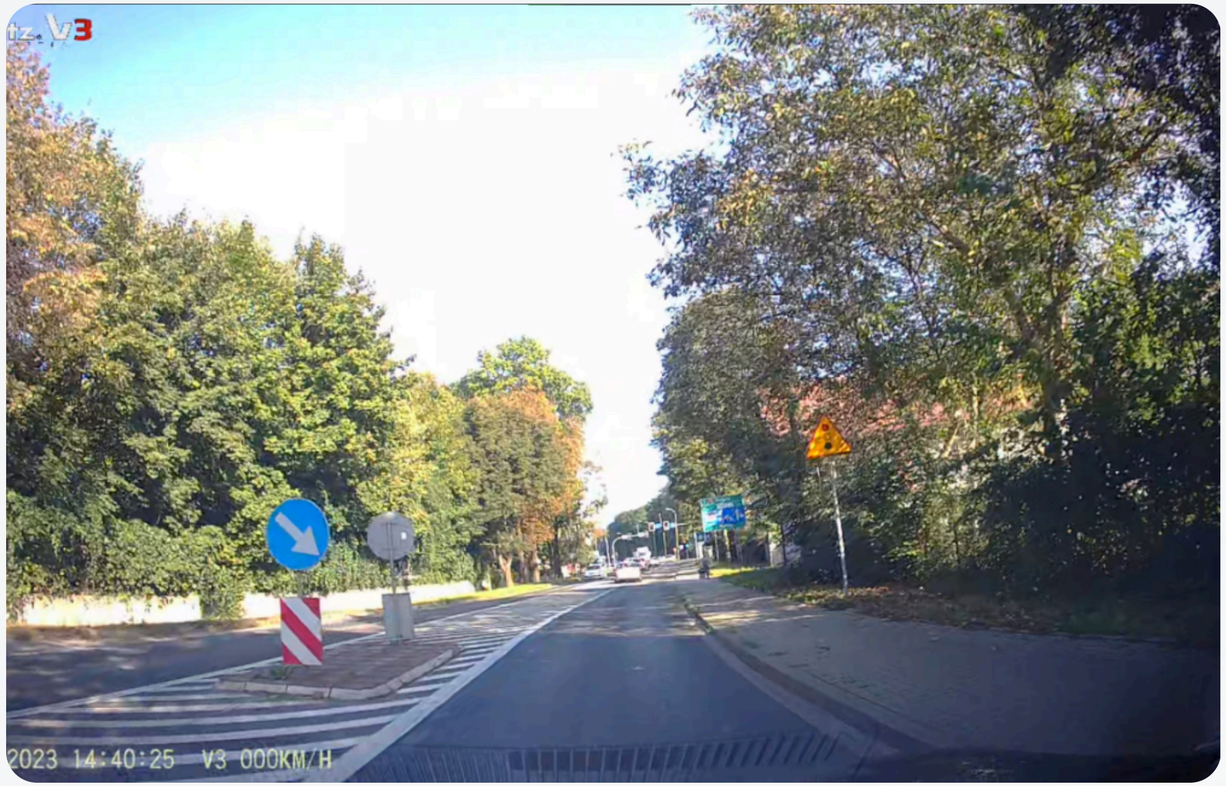
3RD PREDICTION EXAMPLE

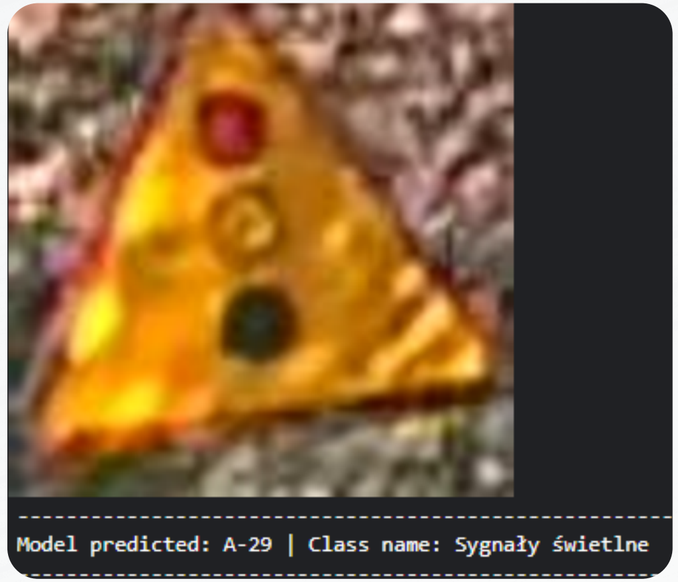
4TH PREDICTION EXAMPLE

5TH PREDICTION EXAMPLE

6TH PREDICTION EXAMPLE

7TH PREDICTION EXAMPLE





FINAL RESULTS ON CAR CAMERA PHOTOS

?%

SUM OF SIGNS DETECTED CORRECTLY	ALL	ACCURACY
43	53	~81%

Lack of detection is the most common problem - my yolov8 model is learned on a medium distance character dataset. Because of this, it sometimes does not cope with the detection of signs at a distance of 100 to 200 meters

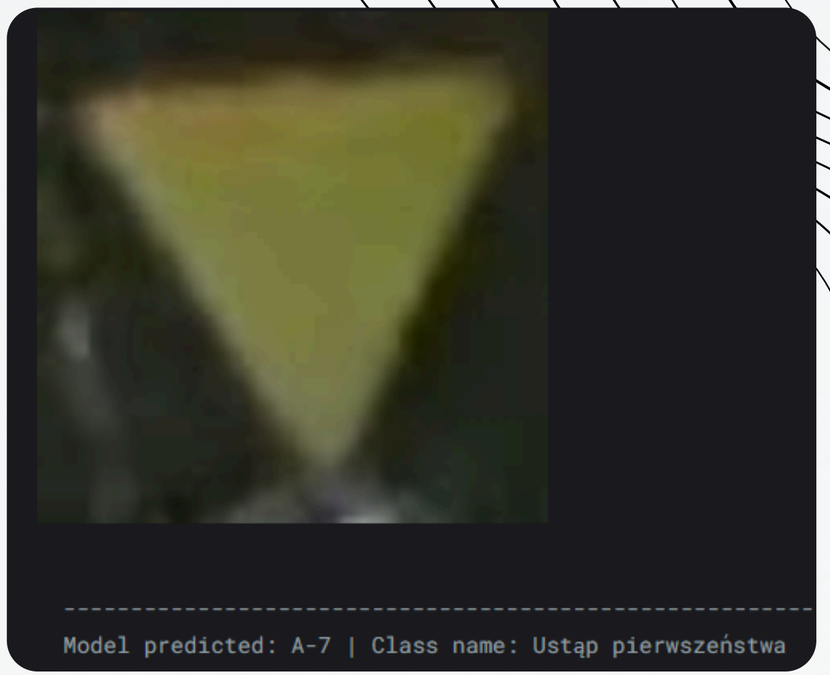
CORRECT CLASS PREDICTIONS AFTER DETECTIONS	ALL	ACCURACY
38	41	~93%

The model does a great job of classifying road signs. It is wrong only with very similar classes, e.g. C-5 and C-9 (due to different shooting angle)

FULL CORRECT DETECTIONS THEN PREDICTIONS	ALL	ACCURACY
17	30	~57%

Full prediction means the detection of all signs and the correct classification of each of them. It often happens that 4/5 of the signs have been well detected and classified, but such attempts do not count towards full prediction

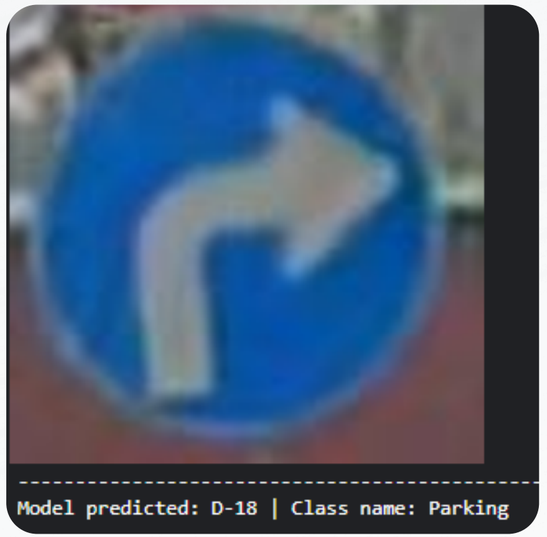
EXAMPLE OF “BAD” DETECTION



2ND EXAMPLE OF “BAD” DETECTION

Perfect example of no-detection while road sign is too far away. The reason is propably poor quality of the sign or background mixing up some shapes of real sign

3RD EXAMPLE OF “BAD” DETECTION



FULL CORRECT PREDICTIONS AFTER DETECTIONS ON SIGNS WITHIN 5m - 50m	ALL	ACCURACY
25	30	~83%

When the signs are reasonably close to the car, the weather conditions are good and the sign is well lit, my hybrid model performs at 83%

CONCLUSIONS

The main goal was to create a tool for recognizing Polish road signs and it is achieved with high % correct predictions on cropped sign images

My detection model (used to crop signs from original image) could have been trained better, by adding some “bad quality” images to the dataset

In the end, I feel that my project achieved the required target, although it could certainly have been brought to a slightly better percentage of success