

# CONTENT

**01**

PROJECT GOALS

**02**

APPROACH

**03**

ENCOUNTERED PROBLEMS

**04**

FINAL MODEL

**05**

EFFECTS & RESULTS

**06**

SUMMARY

# 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, we really wanted them to be Polish road signs, because we wanted to then finally test the model on our car images.

## SIGN DETECTION

We 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, we 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. We 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 we can learn from



## ENCOUNTERED PROBLEMS

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



# UNBALANCED DATASET

We 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 us to decision of choosing option 4 with following steps:

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

FLOATING NUMBER  
OF IMAGES PER  
CLASS

We 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

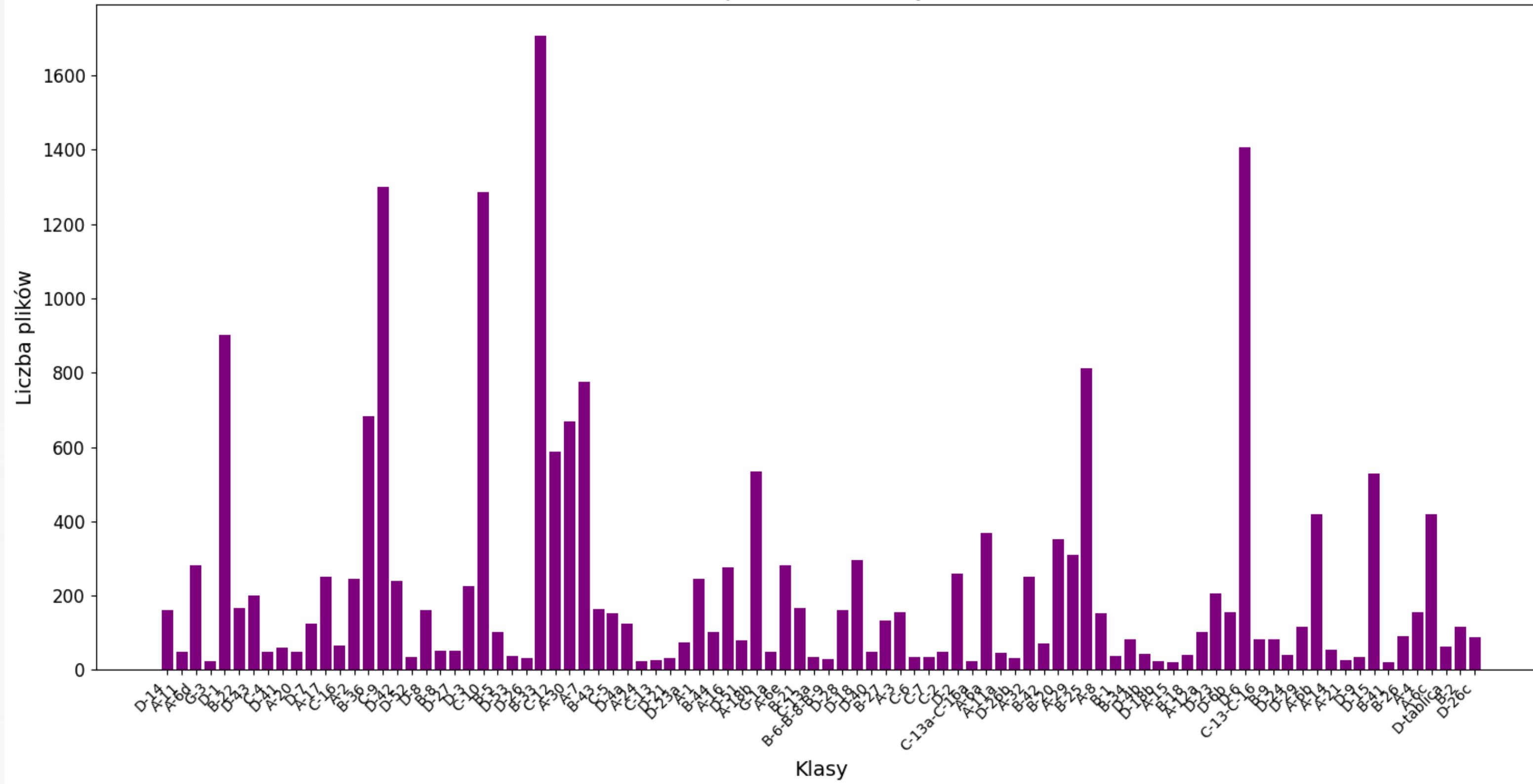
In the remaining classes, we applied a truncation of too many photos to 300, and in classes where photos were missing, we performed augmentation. Operations of gentle rotation, color changes, blurring, shifting and brightness changes helped produce photos up to the designated number of 300 per class

AUGMENTATION

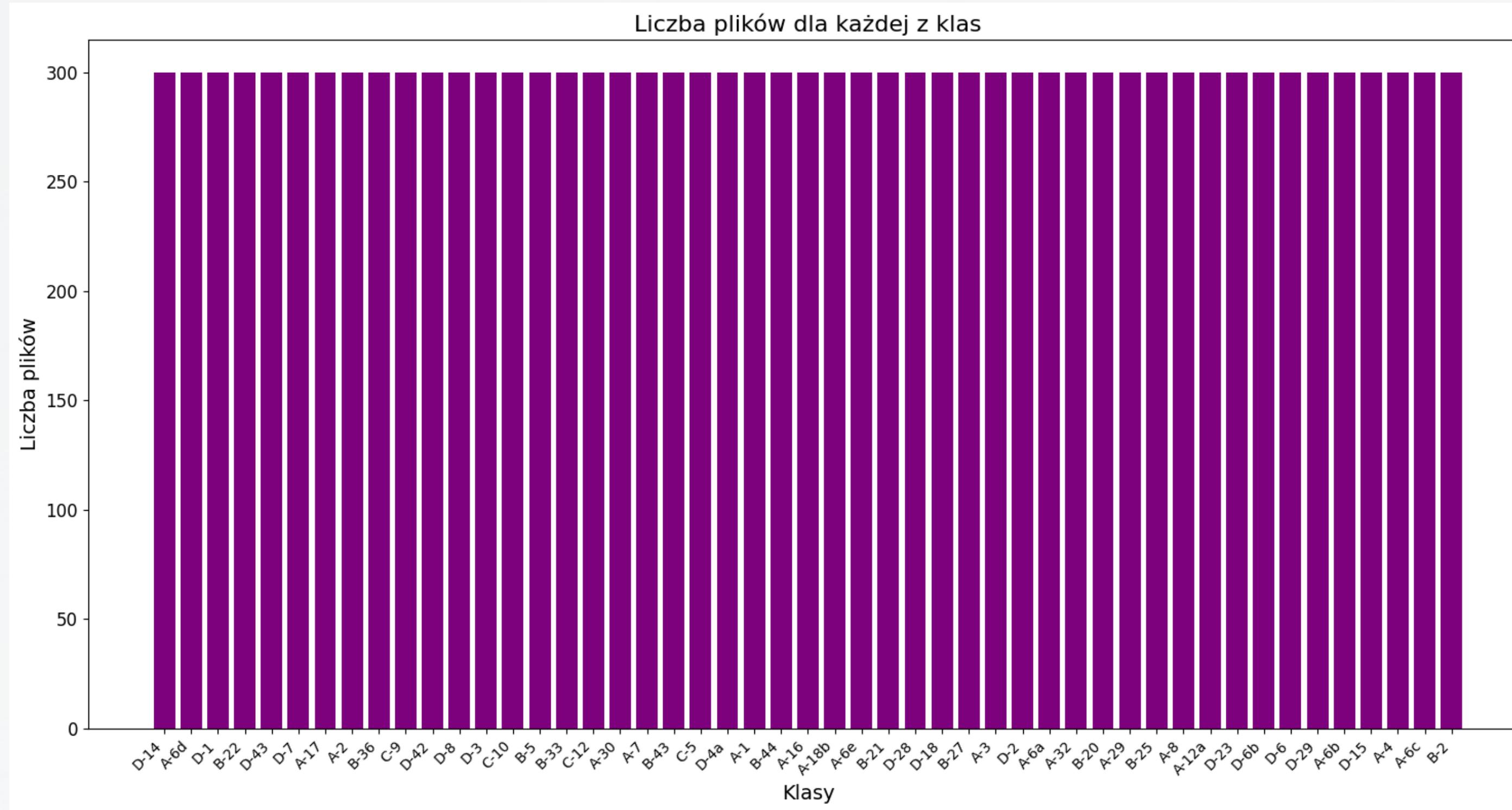
# FLOATING NUMBER OF IMAGES PER CLASS



Liczba plików dla każdej z klas



# AFTER REMOVAL & AUGMENTATION



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

# EXAMPLE OF DATA SAMPLES PER CLASS



# SHAPING MAIN CNN MODEL

We chose to create our own model based on our knowledge from the lectures and any material from the Internet. During the creation process, we did a lot of testing, while building an intuition that we did not have before

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

## BASIC CNN MODEL

We 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

Our model was overfitting. We applied L2 regularization, dropout after each convolutional layer and in the fully connected layer. We 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),

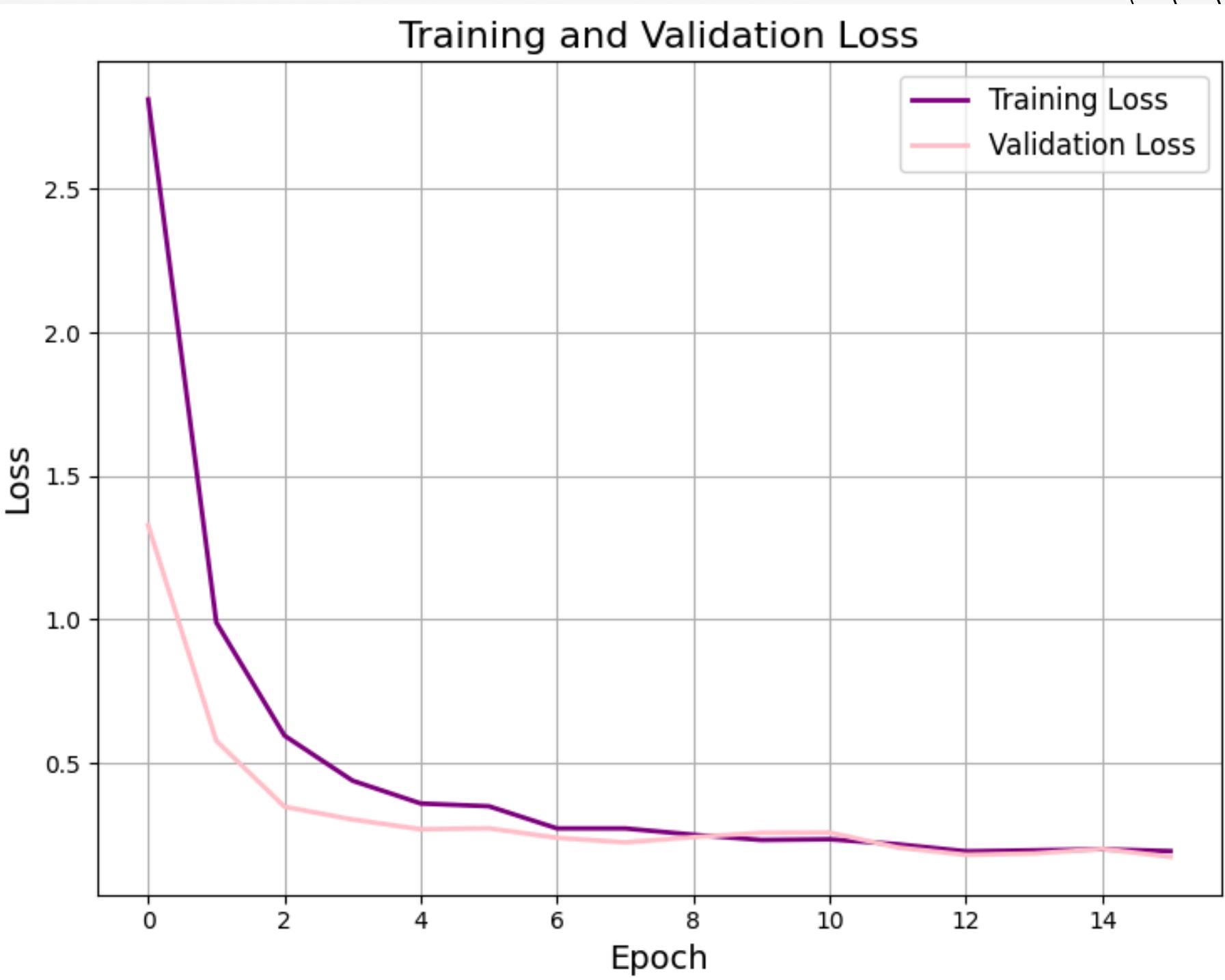
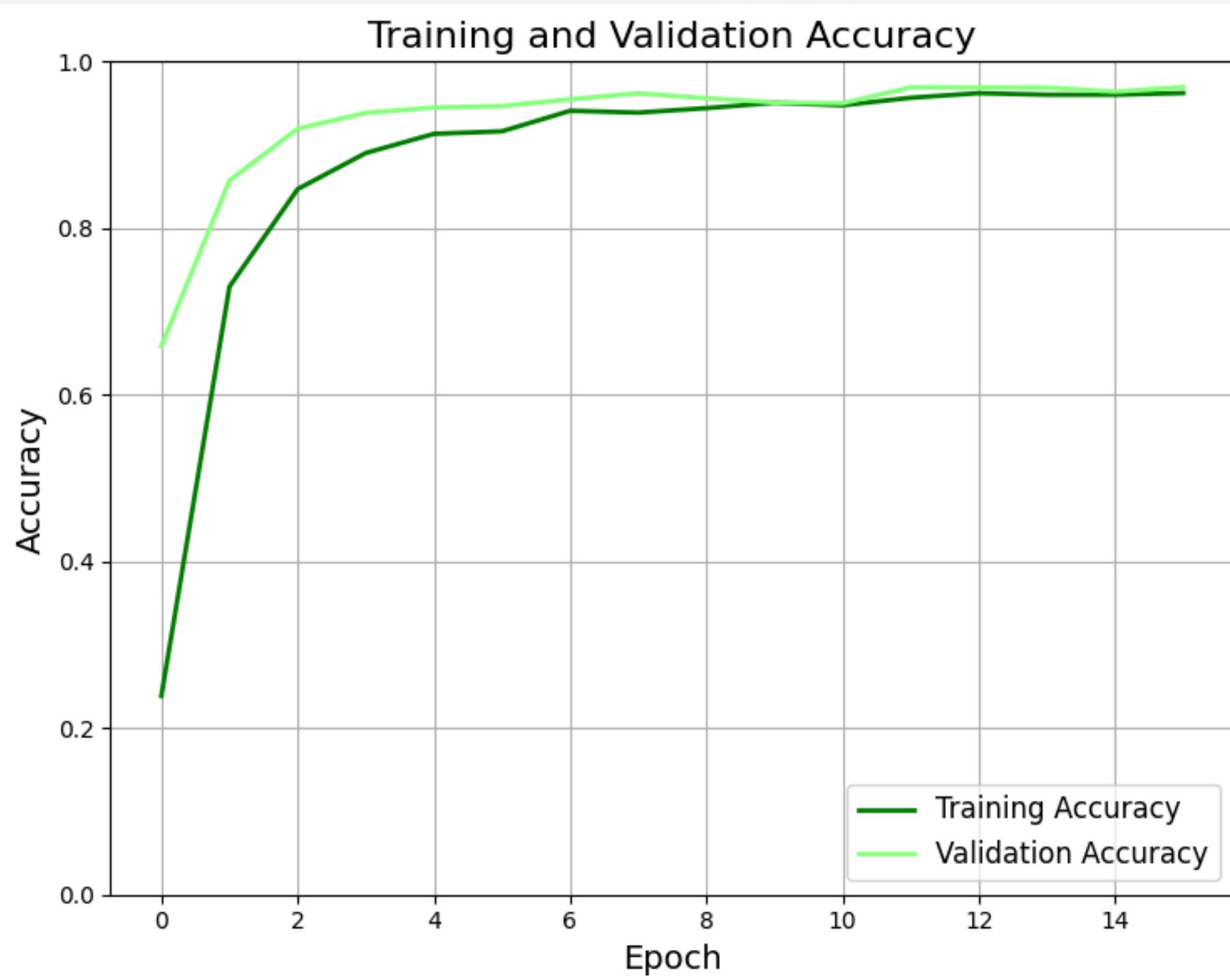
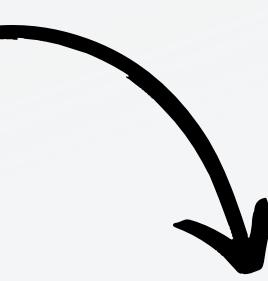
    Flatten(),
    Dense(294, activation='relu'),
    Dropout(0.2),
    Dense(49, activation='softmax')
])
```

# HYBRID MODEL SUMMARY AND STATISTICS

95%



# ACCURACY AND LOSS CURVES OF CNN MODEL



# STATISTICS



Ultimately, our 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

CNN MODEL  
ACCURACY



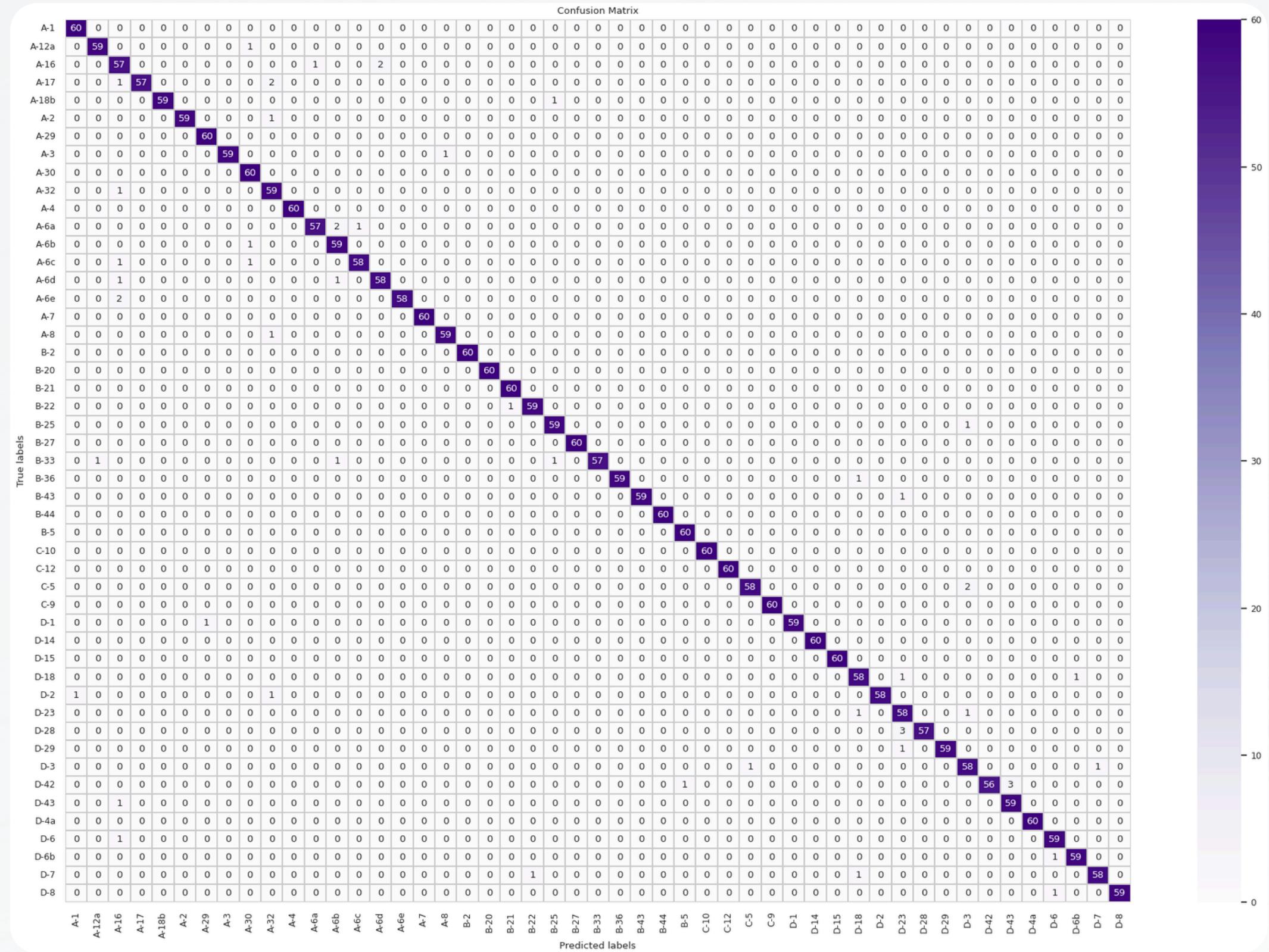
On average, the CNN model was able to achieve about **95%** success in recognizing sign classes from cropped images



**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  
Ustęp pierwszeństwa

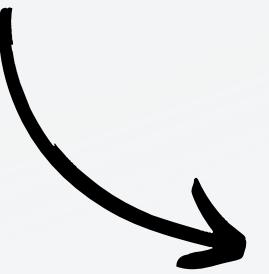


D-6  
Przejście dla pieszych



B-36  
Zakaz  
zatrzymywania się

# IN-CODE PROOF



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

predict_with_hybrid_model(random_jpg_file_path)
```



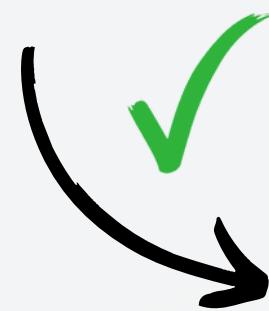
# DETECTION & PREDICTION



# DETECTION & PREDICTION



# DETECTION & PREDICTION

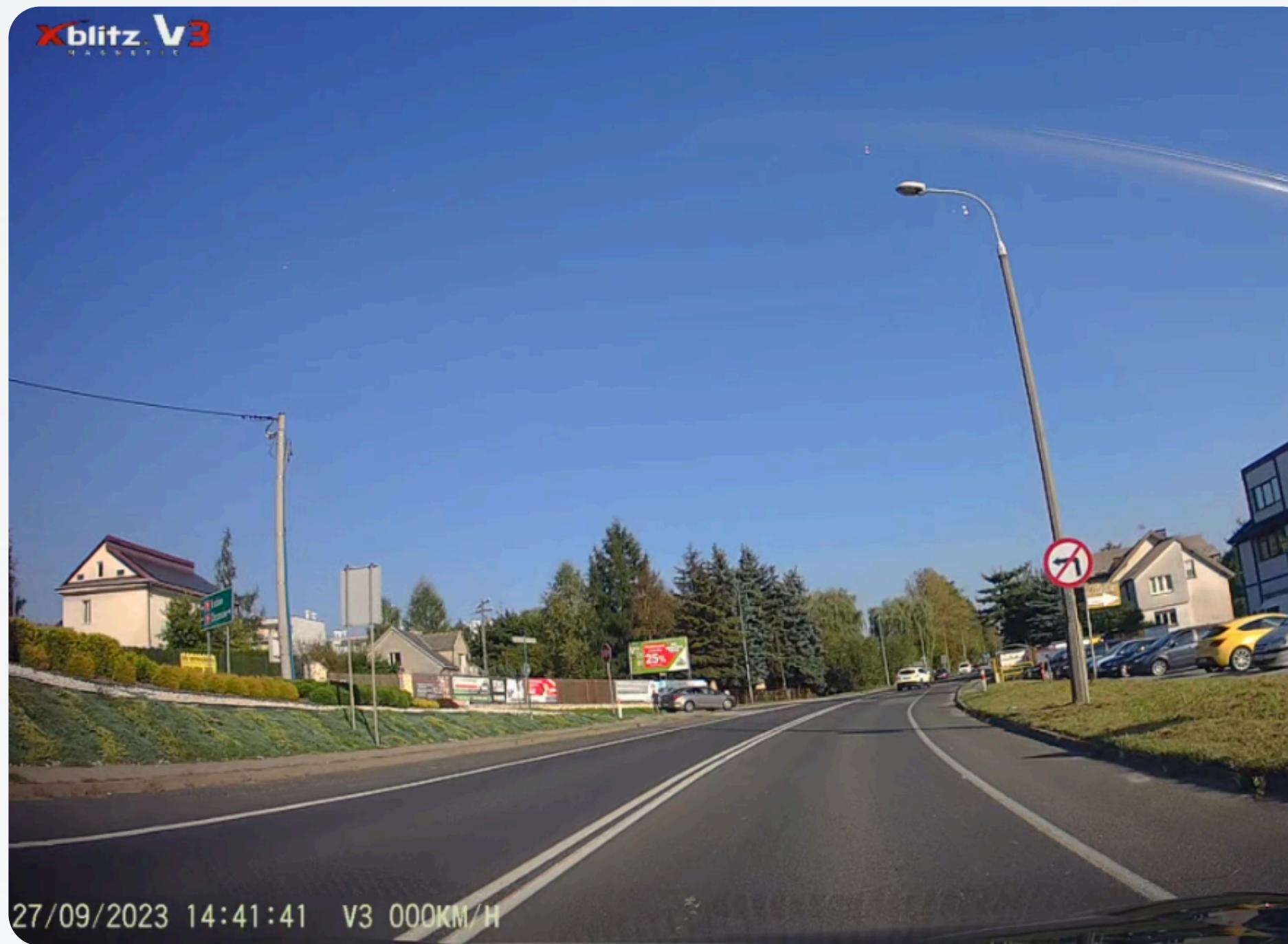


# DETECTION & PREDICTION

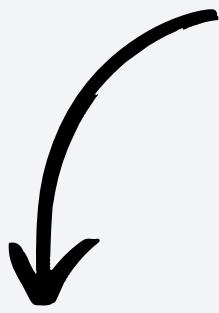


Model predicted: B-36 | Class name: Zakaz zatrzymywania się

## 2ND PREDICTION EXAMPLE



# 3RD PREDICTION EXAMPLE

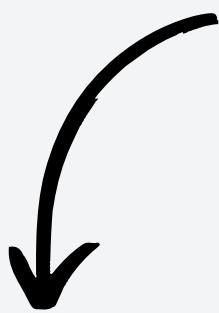


Model predicted: A-16 | Class name: Przejście dla pieszych

# 4TH PREDICTION EXAMPLE

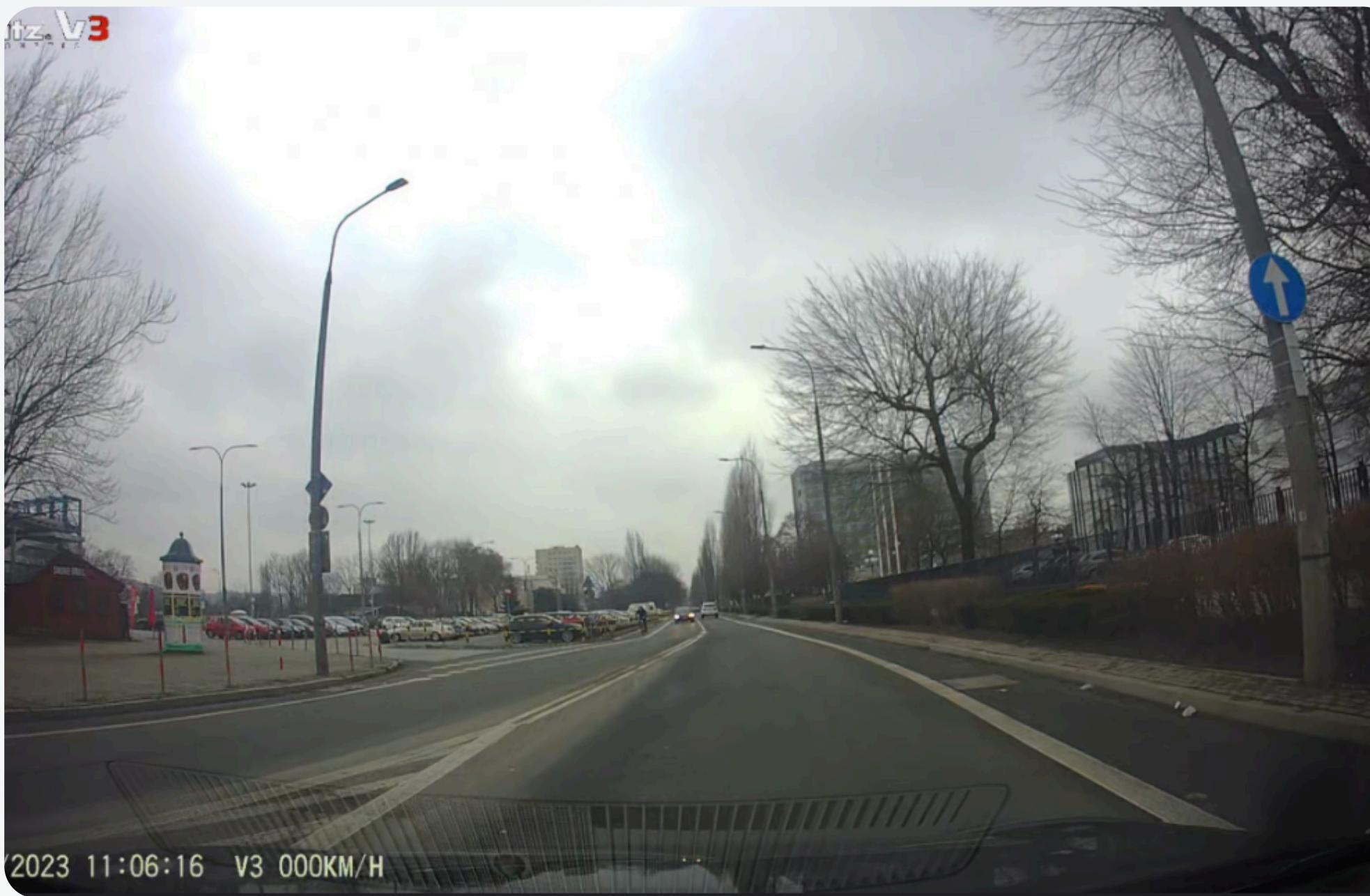


# 5TH PREDICTION EXAMPLE

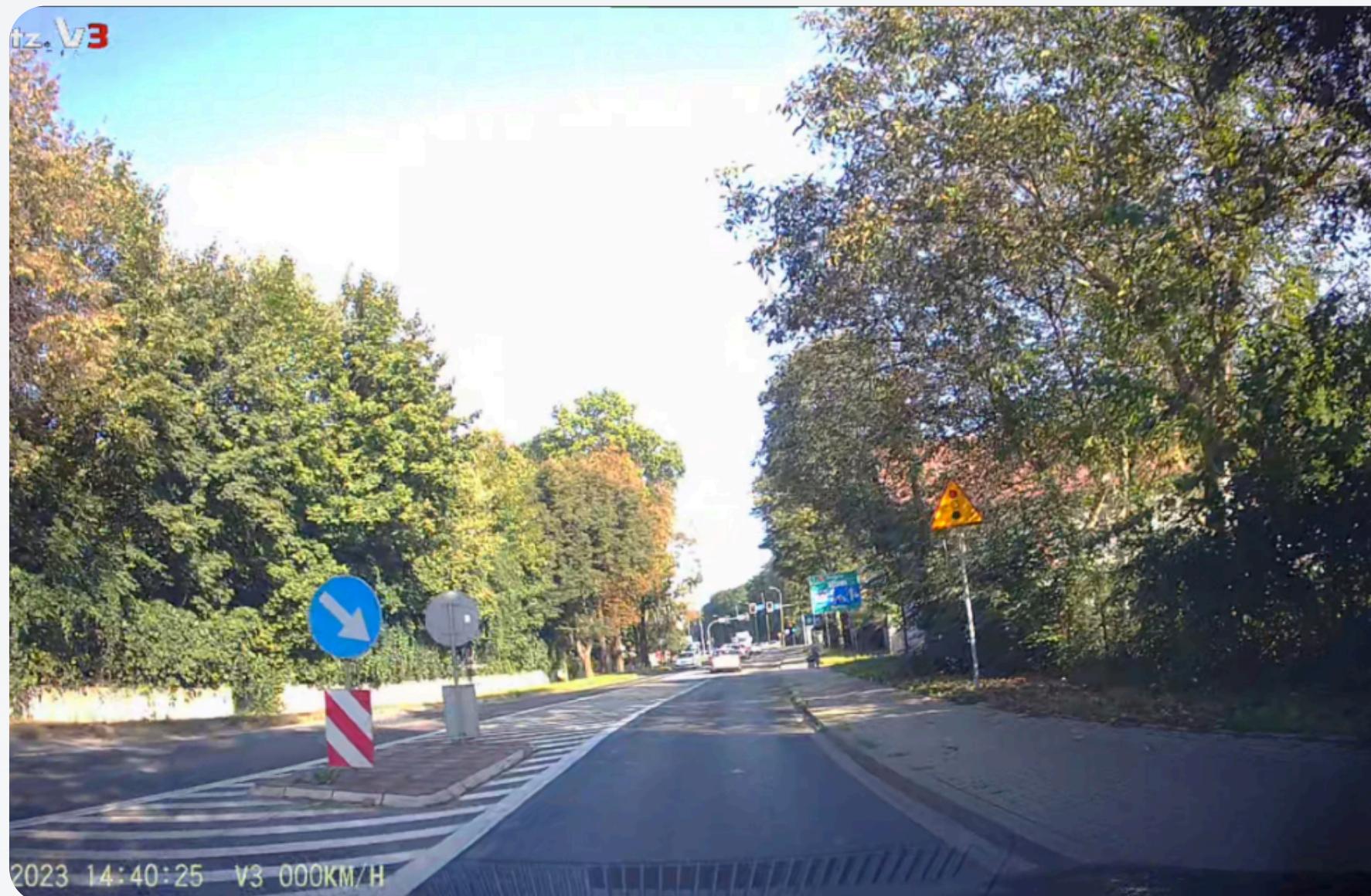


Model predicted: B-2 | Class name: Zakaz wjazdu

# 6TH PREDICTION EXAMPLE



# 7TH PREDICTION EXAMPLE



Model predicted: C-9 | Class name: Nakaz jazdy z prawej strony znaku



Model predicted: A-29 | Class name: Sygnały świetlne

**FINAL RESULTS  
ON CAR CAMERA  
PHOTOS**

**?%**



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

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

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

Lack of detection is the most common problem – our 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

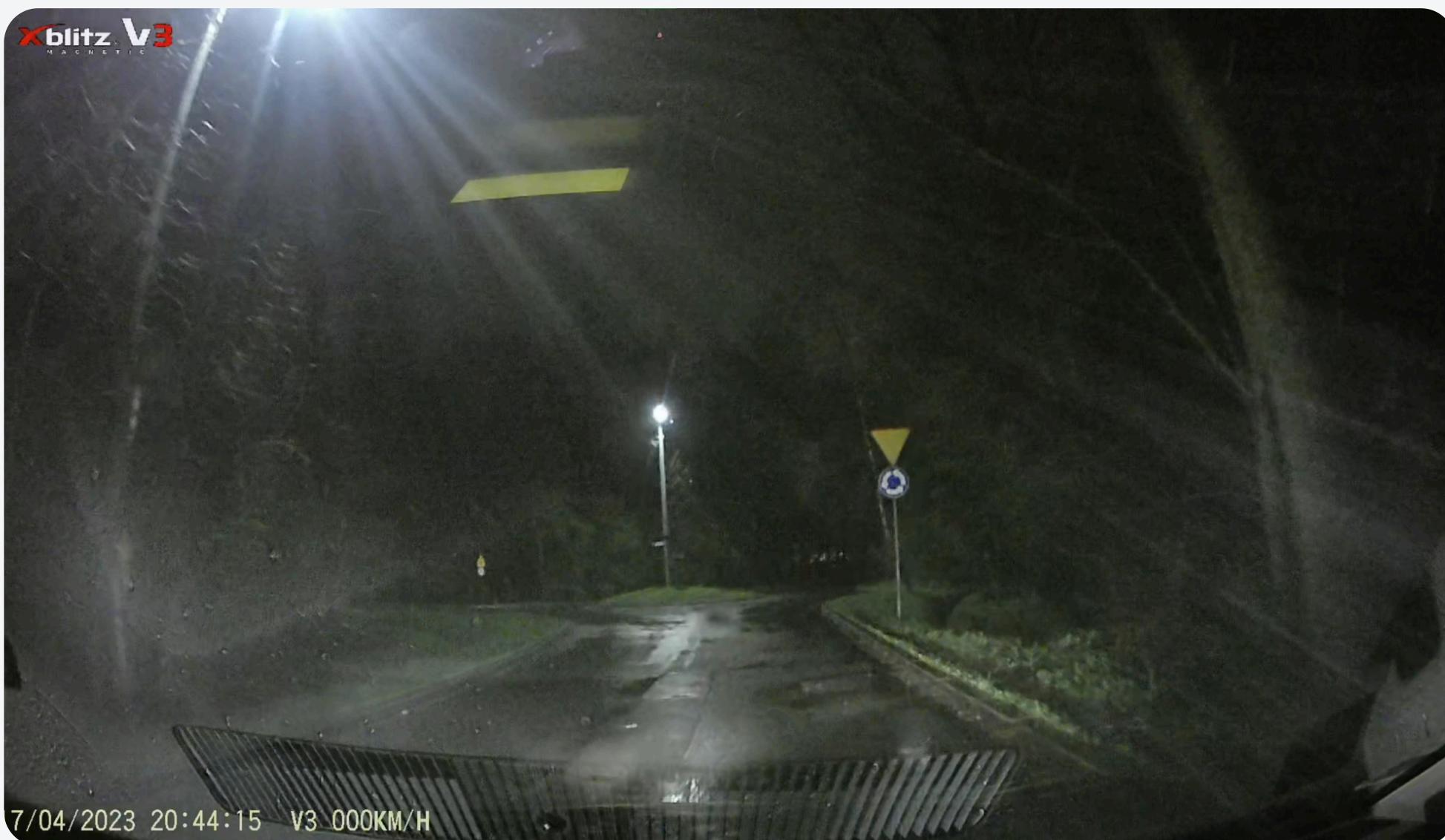


The model does a great job of classifying road signs. It is wrong only with very similar classes, e.g. C-5 and C-9 (due to different shooting angle)



Full 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

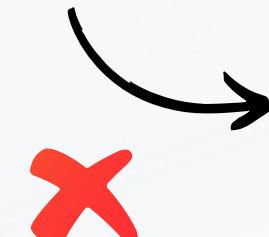
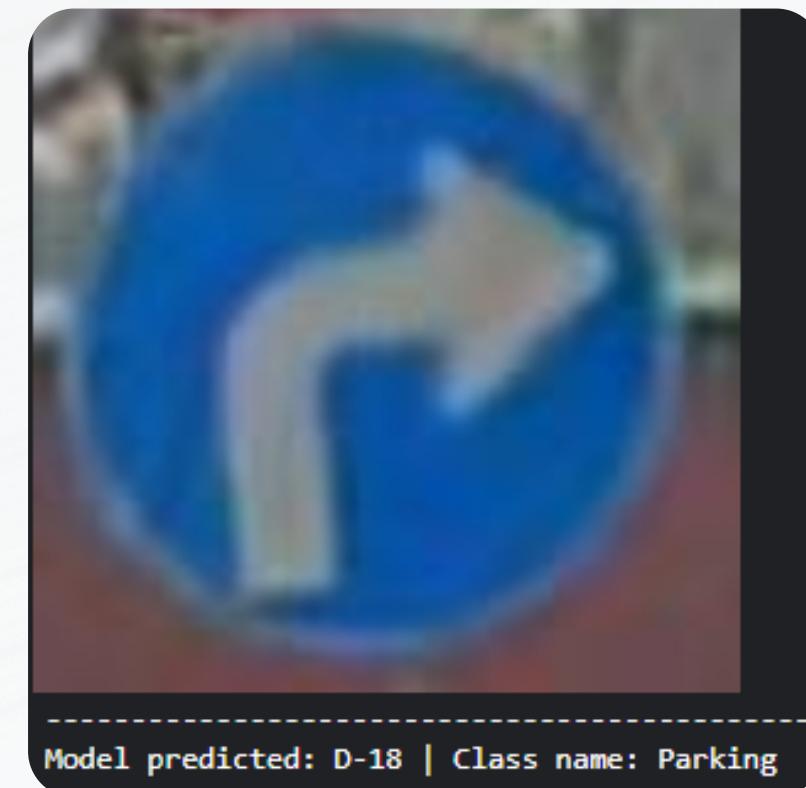


## 2ND EXAMPLE OF “BAD” DETECTION



Perfect example of no-detection while road sign is too far away. The reason is probably 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<br>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, our 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



Our 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, we feel that our project achieved the required target, although it could certainly have been brought to a slightly better percentage of success

