# License Plate OCR

## Computer Vision Project

Made by: Tomaily Tatiana, Gorelskii Roman, Sergazin Iskander
Supervisor: Kopylov Ivan

## 01 Introduction

Description. Application. Problem definition. Technology stack.

## 02 Tasks Division

Project stages. Tatiana. Roman. Iskander.

## 03 Review of Existing Solutions

Approaches already used. Reason of the proposed methods chosen.

## 04 Data Preparation

Description of the dataset and Visualization. Data preprocessing.

## 05 Model Development

Neural network. Why the approach was chosen. Hyperparameters.

## 06 Implementation and Training

Code & Environment. Training Process.

## 07 Model Evaluation

Metrics. Testing on new data. Error interpretation.

## 08 Deployment & Demonstration

Model usage in production. API development. Interactive prototype

## 09 Conclusions & Further Perspectives

Key results. Unsolved problems. Ways to improve the model.

## 10 References

# 01
# Introduction

## Description

**Manual controlling of vehicles** which are allowed to enter the area is often **slow, error-prone, and inconvenient.** Our project **solves this problem by developing** an automatic license plate recognition system that detects a car's license plate and checks if it exists in a **pre-approved database.**

## Application

Potentially, it can **be integrated with automated gates,** allowing only authorized vehicles to pass through them. Potential users include **residential complexes, corporate offices, and event organizers seeking efficient access control.**


# Problem definition

The main goal of this project is to **automatically recognize vehicle license plate numbers from camera footage** or images and **check whether the detected number exists in a predefined access list.**

# Technology stack

- Python
- OpenCV
- Telegram Bot Api
- OCR method implementation

# 02
# Project stages
# Tasks Division

| | Roman | Iskander | Tatiana |
|---|---|---|---|
| Review of existing solutions and data collection | ✓ | ✓ | ✓ |
| Bot and Database implementation | ✓ | | |
| Model deployment, training and metrics improvement | | ✓ | |
| Matching plates with database, returning a response | | | ✓ |
| Prototype development | ✓ | ✓ | ✓ |

# 03
# Review of Existing Solutions

## Approaches already used

- **Traditional methods** using OpenCV with **contour** detection or **edge** detection
- Deep learning-based **object detection models like YOLO**
- Deep learning-based CRNN (**Convolutional Recurrent Neural Networks**) models
- **Optical character recognition** (OCR) – include Tesseract OCR (open-source engine), EasyOCR

## Our method

Lightweight and fast license plate OCR pipeline inspired by the Fast Plate OCR approach

- **Efficiency**: optimized for real-time processing
- **Simplicity**: easy to deploy and maintain
- **Accuracy in Diverse Conditions**: better generalization
- **Scalability**: allows future enhancements

# 04
# Data Preparation

# Description of the dataset and Visualization

**Source:** we used an open-source dataset from the Fast Plate OCR GitHub repository.

This dataset contains images of vehicles with visible license plates.


**Quantity:** 2874 for training and 764 for validation

**Format:** images in .jpg format, Argentina plates

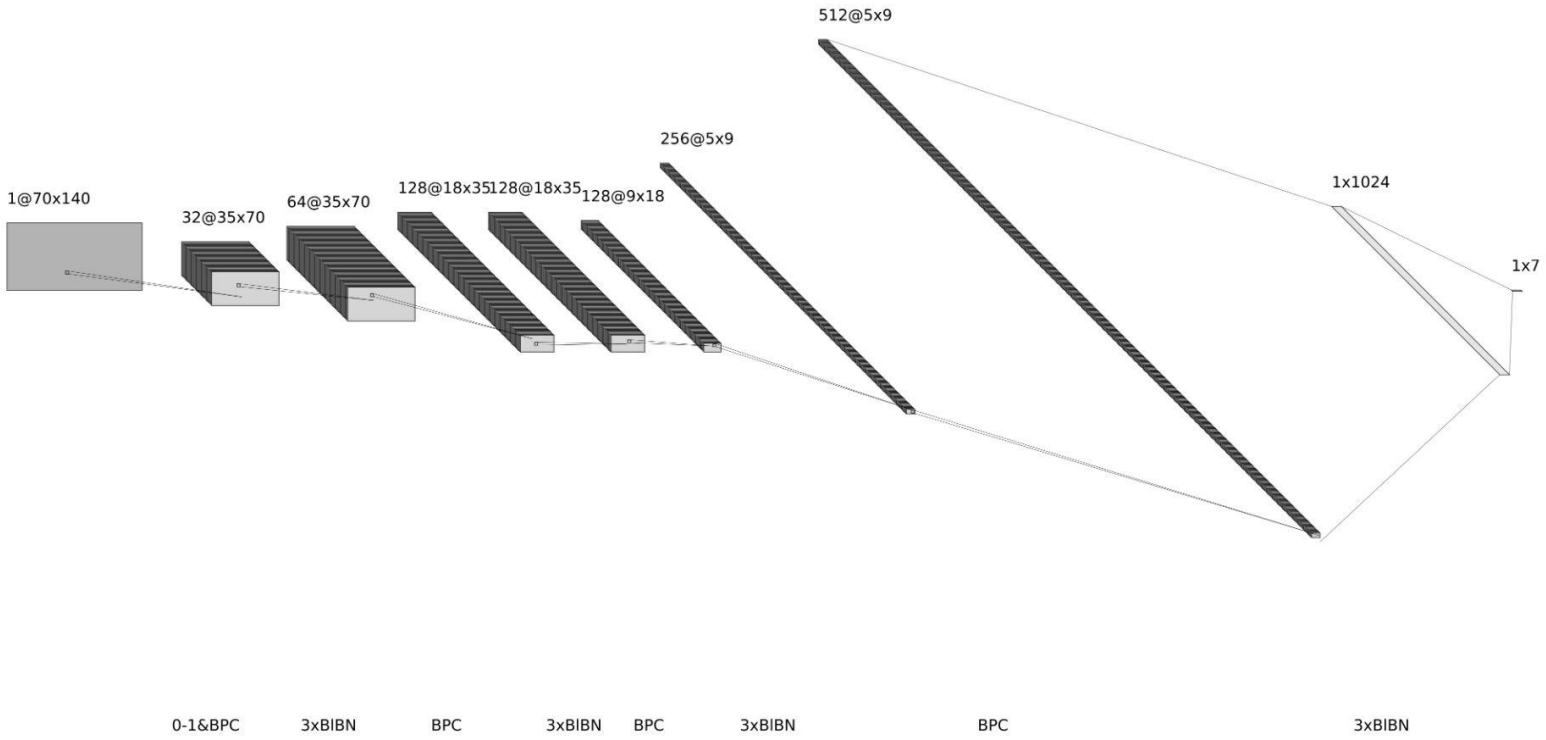
# Data preprocessing

The augmentation technique is used to get better predictions

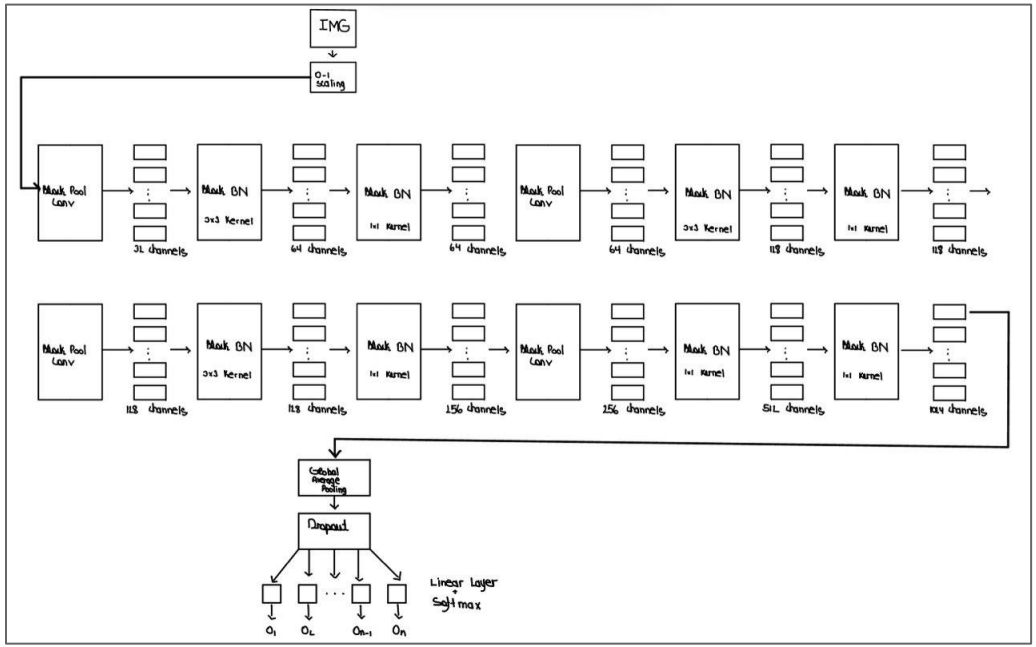

Data before and after augmentation

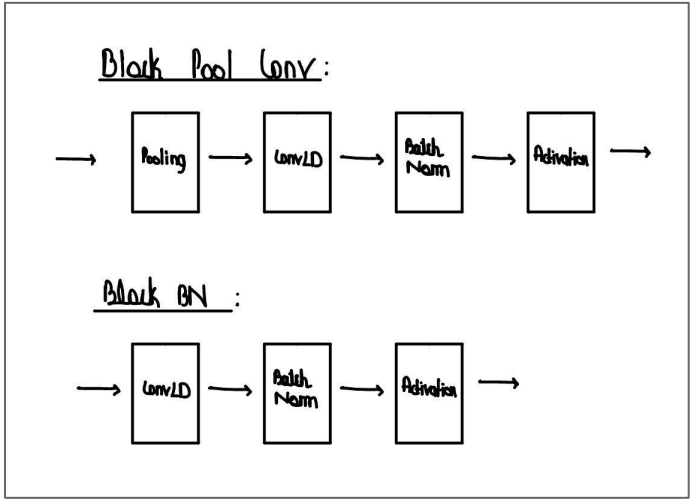
# 05
# Model Development

# Neural Network Architecture



# Neural Network Architecture (2)




https://github.com/ankandrew/fast-plate-ocr/blob/master/fast_plate_ocr/train/model/models.py

# Why the approach was chosen?

## Fast Plate OCR

- **Lightweight** model (Throughput 400 frames/s)
- Can be **trained from zero** (8hrs 2500 img)
- **Pre-trained** models (argentinian/european/world) plates

# Strategy & Hyperparameters

## early_stopping

Stop training when val_plate_acc does not improve over 100 epochs

## epochs

750

## reduce_lr

0.5 reduction in LR when no improvement in val_plate_acc

## Config File

7 characters predicted. Input image 70×140

## batch_size

128

## ReLU & MaxPooling

Activation f’n & Pooling Layer (can be chosen)

# 06
# Implementation and Training

# Code & Environment

- Python 3.10
- requirements.txt
- GitHub

**Github repositories:**

LicensePlateDetection (Our project) - https://github.com/RomanGorelsky/CV_Project_Plates

FastPlateOCR - https://github.com/ankandrew/fast-plate-ocr

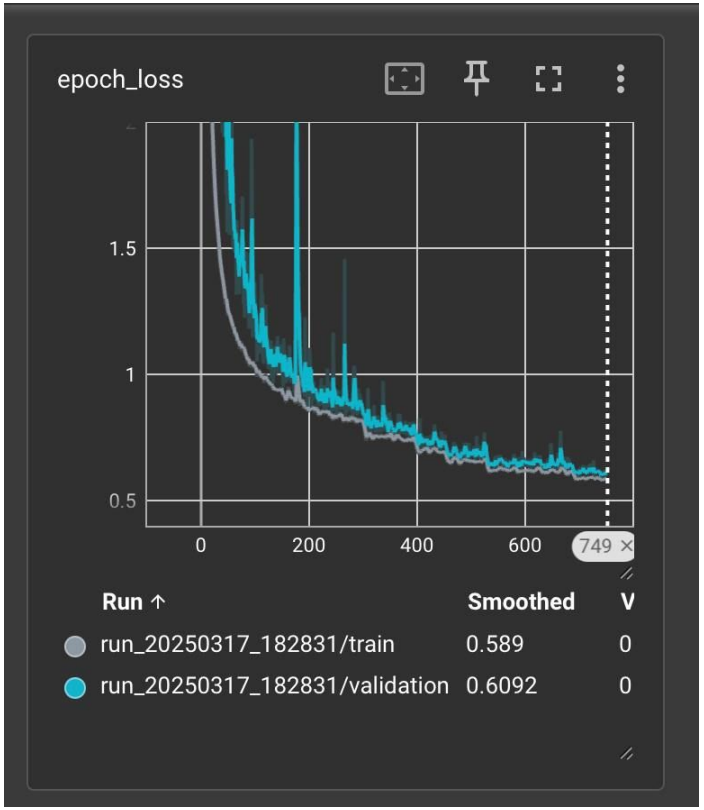
# Training Process

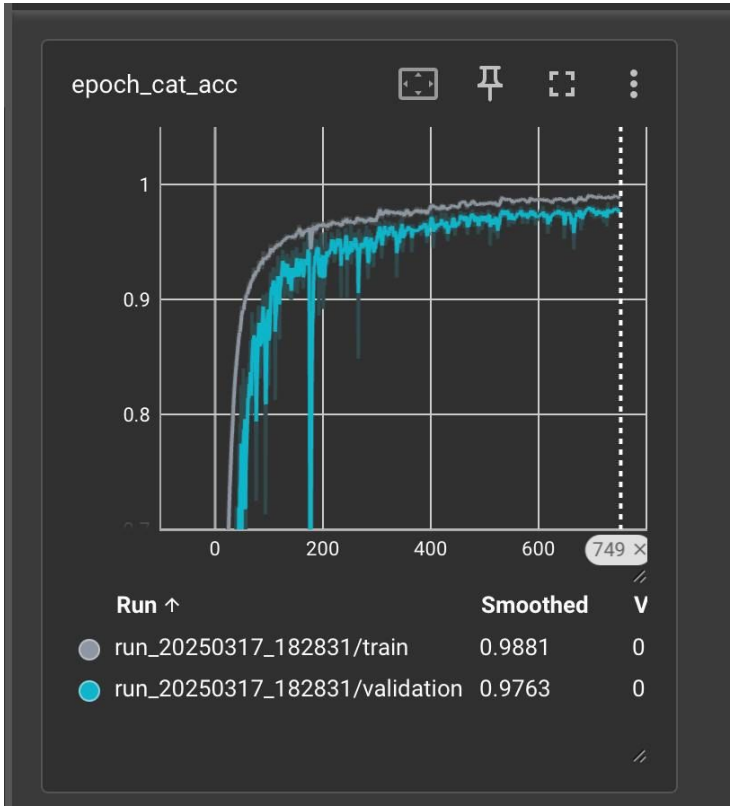
## Training

- **Loss:** Mean categoricalCrossEntropy over each license plate digit
- **Optimizer:** Adam

$$\mathcal{L} = -\sum_{i=1}^{C} y_{\text{true},i} \cdot \log(y_{\text{pred},i})$$

# Training Process (2)



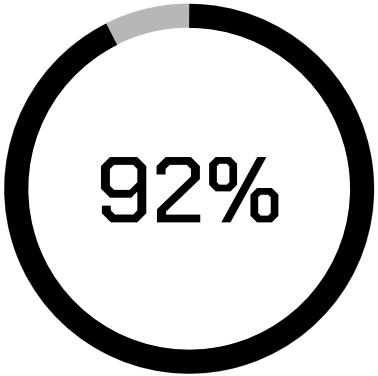



# 07
# Model Evaluation

# Metrics

| | Explanation |
|---|---|
| **Accuracy** | Plate accuracy over whole dataset |
| **Plate Precision** | Average character precision within incorrect plates |
| **Top 3-k** | Whether correct character in top-3 model predictions |

# Testing on data




Accuracy

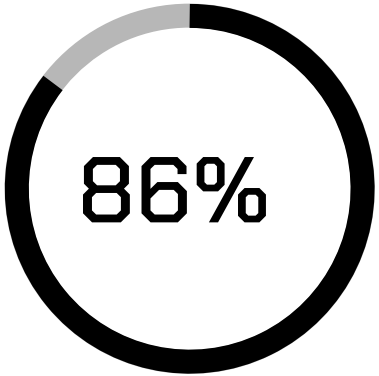
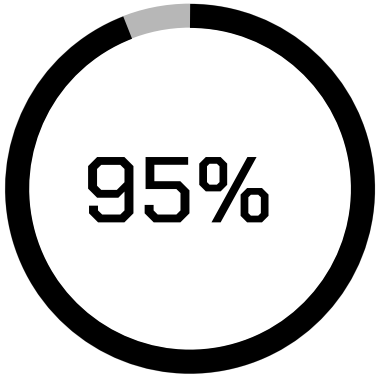
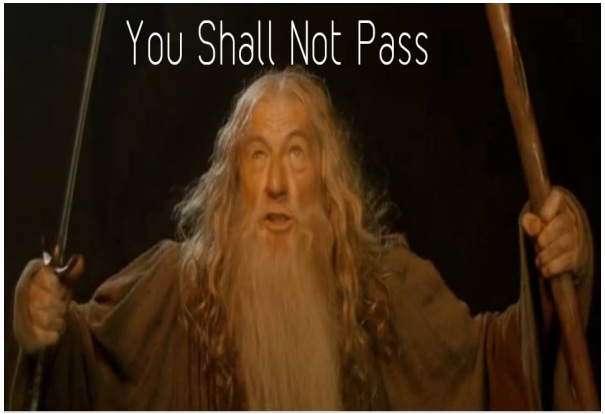



Plate precision

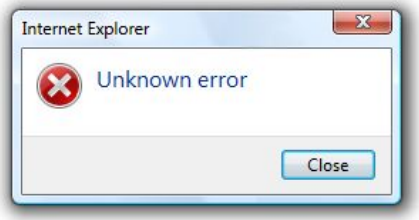



Top 3-k

# Error interpretation


Model

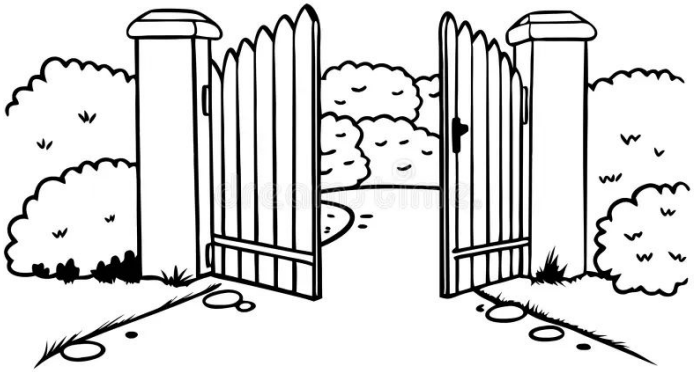



- **Confuses letters** (Q/O, U/V)
- **Learns distribution** of data (LLLDDLL)
- **Poor performance** on plates from a **different** distribution (**Russian** plate dataset)

# 08
# Deployment and Demonstration

# Model usage in production

The model offers the possibility to implement the automatic opening of gates and barriers when the car arrives. The realisation consists in the **possession of the camera** located near the gate and **the database** in which license plate collection is stored. Such systems can be installed at various locations for personal or commercial use.



# API development

In order for a driver to have the possibility to add their plate for granted access the API with **telegram bot** was developed:

- Library “pyTelegramBotAPI“ was used
- The bot was made **asynchronous** allowing multiple users to utilise it at once
- With the prompt “Plate: …” the license plate is added to the database
- **Duplicates** are checked and not added
- All the plate numbers are stored in uppercase

The plate number is added into the database!

# Interactive prototype

# 09
# Conclusions & Further Perspectives

## Key results

- Project dataset
- Fast **Plate OCR** methodology **training**
- **Telegram** bot deployment
- **Interactive prototype** implementation

## Unsolved problems

- **No** model **versatility**
- Simulation and visualisation of **gate opening process**

## Ways to improve the model

- Train on license plates **from different countries**
- Grant access for **specific time length**
- **Extraction** of plates in the **wild**
- Implement the check of **similar plates combinations**
- Implement API ability to **delete specific plates**
- Integrate model into API to **add plates from photo**

# 10
# References

Fast Plate OCR:
https://github.com/ankandrew/fast-plate-ocr
Requirements:
https://drive.google.com/file/d/1p0sDDkpPJjHkerbEo5BDa0c3PVdW_UaJ/view?usp=sharing
Project's GitHub:
https://github.com/RomanGorelsky/CV_Project_Plates.git

# Thanks!

Do you have any questions?