Ministry of Science and Higher Education of the Republic of Kazakhstan
L.N. Gumilyov Eurasian National University

Faculty of Information Technology
Department of Information Systems

midterm number plate recognation

Done by: Akbayan, Aruzhan, Inkara

INTRODUCTION

WHAT IS NPR?

NPR, also known as License Plate Recognition (LPR), is a computer vision technology that uses image processing and machine learning to automatically identify and read vehicle license plates.
It combines optical character recognition (OCR) with video or image capture to detect and extract alphanumeric text from license plates.

HISTORICAL ROOTS

The Automatic Number Plate Recognition (ANPR) was invented in 1976 at the Police Scientific Development Branch in the UK. However, it gained much interest during the last decade along with the improvement of digital camera and the increase in computational capacity. It is simply the ability to automatically extract and recognition a vehicle number plate’s characters from an image. In essence it consists of a camera or frame grabber that has the capability to grab an image, find the location of the number in the image and then extract the characters for character recognition tool to translate the pixels into numerically readable character.

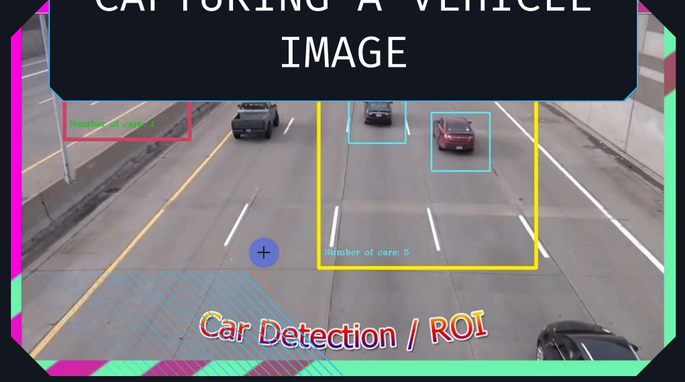
MODERN IMPORTANCE

ANPR can be used in many areas from speed enforcement and tool collection to management of parking lots, etc [1]. It can also be used to detect and prevent a wide range of criminal activities and for security control of a highly restricted areas like military zones or area around top government offices.

HOW NPR WORKS?

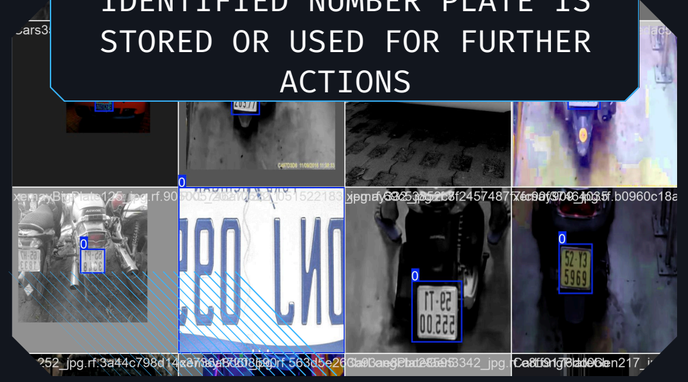
The ANPR system works in three steps, the first step is the detection and capturing a vehicle image, the second steps is the detection and extraction of number plate in an image. The third section use image segmentation technique to get individual character and optical character recognition (OCR) to recognize the individual character with the help of database stored for each and every alphanumeric character.

THE DETECTION AND CAPTURING A VEHICLE IMAGE



THE DETECTION AND EXTRACTION OF NUMBER PLATE IN AN IMAGE

RECOGNIZES AND CONVERTS CHARACTERS INTO TEXT. IDENTIFIED NUMBER PLATE IS STORED OR USED FOR FURTHER ACTIONS



DATA PREPARATION

METHOD

MODEL TRAINING

TEXT RECOGNITION (OCR)

To create automatic license plate recognition mf used Python. We will look at detecting and reading license plates in video, using YOLO V8 object detector for tracking and EasyOCR for text recognition. The key steps involved, the tools required and useful resources to help realize the process will also be presented.

process

I. DATA PREPARATION

DATA COLLECTION

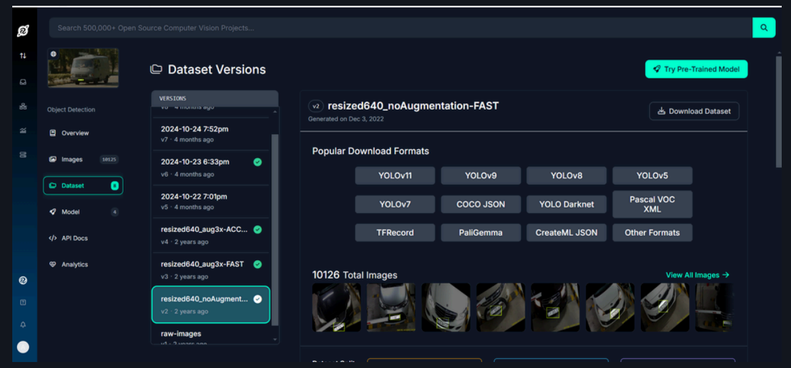
For today's object detection training tutorial, two main methods of data collection are proposed: collecting data on your own or using publicly available datasets. One such dataset is the Open Images Dataset version 7, which includes a huge number of images with millions of annotations across thousands of categories, making it an ideal resource for training object detectors.This project uses a publicly available dataset of data
Collecting and annotating the data is very important for training the object detector, with a variety of methods and categories to choose from. The next step is to continue annotating the data once enough images have been collected.

DATA ANNOTATION

Annotating a dataset is very important when it lacks predefined constraints, especially when using personally collected user data. For such tasks, using a reliable labeling tool such as CVAT can simplify the annotation process. The process of creating annotations for object detection involves drawing bounding boxes around certain objects. The annotator selects a category and circles the object, often maintaining progress to ensure data integrity.Using tools such as VAT can simplify the annotation process and ensure that data is prepared in the format needed to train machine learning models such as YOLOv8.

DATA FORMATTING

The annotation files are named after the corresponding images, with a .txt extension for the YOLO format, and contain class identifiers and data about constraints regarding image size. It is very important to create special directories labeled “images” and “labels” to properly organize the dataset for YOLO V8. It is also recommended to have subdirectories “training” and “validation” to better manage training and validation images.
To prepare the data for YOLO V8, two main directories are created: “images” for the image files and “labels” for the corresponding annotation files, providing each image file with a corresponding .txt label with the same name. The annotations should follow the YOLO format, which includes the class ID, bounding box center coordinates, width and height in relative terms. Once the data is properly structured, you can train the object detector using Python commands or terminal utilities.



The Data Preparation phase involves collecting images, marking them up with tools (e.g., CVAT), and formatting the data in a YOLO-compatible structure. This step is critical for successful model training.

II model training

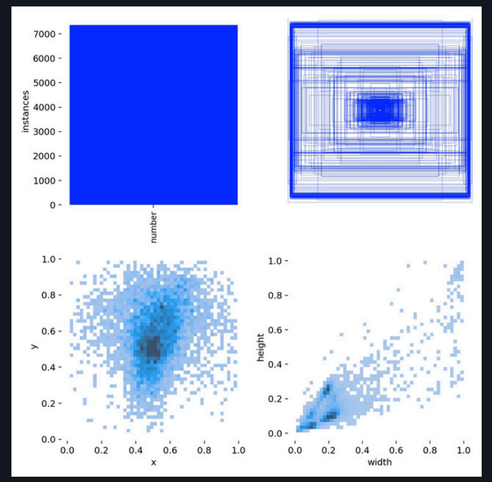
A DATA SET CONTAINING IMAGES OF VEHICLES AND LICENSE PLATES WAS USED TO TRAIN THE MODEL. THE DATA WAS PRE-LABELED, WHICH ALLOWED THE MODEL TO BETTER RECOGNIZE THE TARGET OBJECTS. THE LEARNING PROCESS INVOLVED SETTING UP HYPERPARAMETERS SUCH AS THE NUMBER OF EPOCHS, THE LEARNING RATE, AND THE LOSS FUNCTION. THE TRAINING WAS CONDUCTED ON THE PRE-TRAINED YOLO V8 ARCHITECTURE, WHICH REDUCED TIME COSTS AND IMPROVED DETECTION ACCURACY.AFTER THE TRAINING WAS COMPLETED, THE MODEL WAS TESTED ON A SEPARATE DATASET TO VERIFY ITS QUALITY. THE RESULTS OBTAINED WERE ANALYZED IN TERMS OF ACCURACY, COMPLETENESS, AND F1-MEASURE, WHICH ALLOWED US TO EVALUATE ITS EFFECTIVENESS BEFORE USING IT IN REAL CONDITIONS.

```
                 all       2048       2134       0.96      0.923      0.963      0.658
  0%|          | 0/442 [00:00<?, ?it/s]
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       8/10         0G      1.134     0.5362      1.161          2        640: 100%|██████████| 442/442 [53:36<00:00,  7.28s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 64/64 [05:37<00:00,  5.28s/it]
                   all       2048       2134      0.977      0.925      0.969      0.666
  0%|          | 0/442 [00:00<?, ?it/s]
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       9/10         0G       1.11      0.514      1.144          2        640: 100%|██████████| 442/442 [54:32<00:00,  7.40s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 64/64 [05:37<00:00,  5.27s/it]
                   all       2048       2134      0.976      0.933      0.976      0.687
  0%|          | 0/442 [00:00<?, ?it/s]
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      10/10         0G      1.091     0.4787      1.131          2        640: 100%|██████████| 442/442 [54:23<00:00,  7.38s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 64/64 [05:45<00:00,  5.40s/it]
                   all       2048       2134      0.979       0.94      0.978       0.69
```

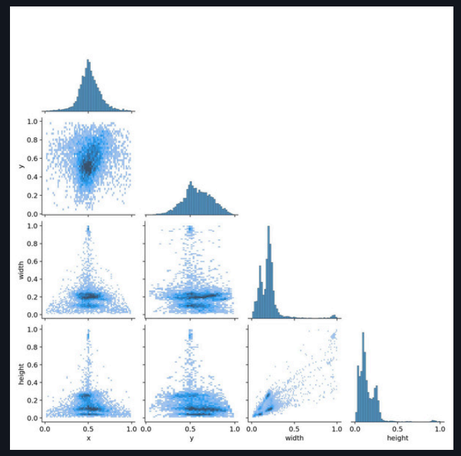
THE EPOCHS

```
10 epochs completed in 10.617 hours.
Optimizer stripped from runs\detect\train2035\weights\last.pt, 6.2MB
Optimizer stripped from runs\detect\train2035\weights\best.pt, 6.2MB

Validating runs\detect\train2035\weights\best.pt...
Ultralytics 8.3.79 🚀 Python-3.11.1 torch-2.6.0+cpu CPU (Intel Core(TM) i3-1005G1 1.20GHz)
Model summary (fused): 72 layers, 3,005,843 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 64/64 [05:02<00:00,  4.73s/it]
                   all       2048       2134      0.978       0.94      0.978       0.69
Speed: 3.4ms preprocess, 134.9ms inference, 0.0ms loss, 0.6ms postprocess per image
Results saved to runs\detect\train2035

Process finished with exit code 0
```

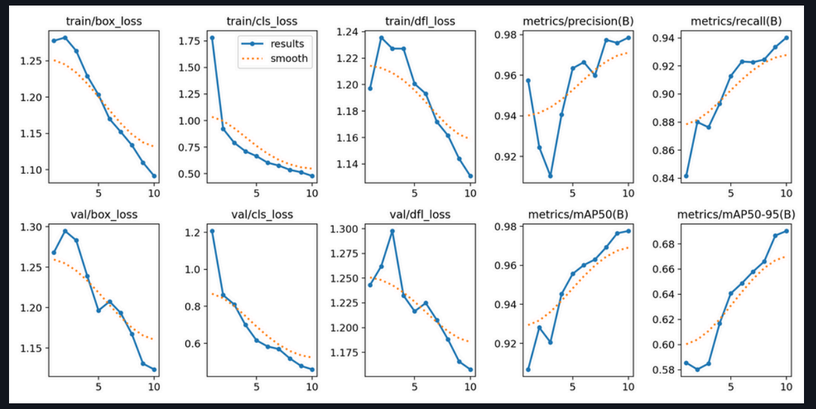
SUCCESSFUL TRAINING



LABELS



LABELS CORRELOGRAM



RESULS

III METHOD

The YOLOv8 (You Only Look Once) algorithm was chosen to implement the license plate recognition system. This algorithm is one of the most modern and effective models for object detection, providing a balance between accuracy and processing speed.

WHY YOLOV8?

- **High-speed operation - YOLO (especially in the nano version) is optimized for real-time operation, which is critical for applications related to video stream processing.**
- **Object recognition accuracy – the model is trained on a COCO dataset and supports recognition of many classes, including cars, motorcycles, buses, and trucks. This makes it possible to reliably detect vehicles on the road.**
- **Flexibility – YOLOv8 supports advanced training on specialized datasets, which allows it to be adapted to specific conditions (for example, to license plates from different countries).**

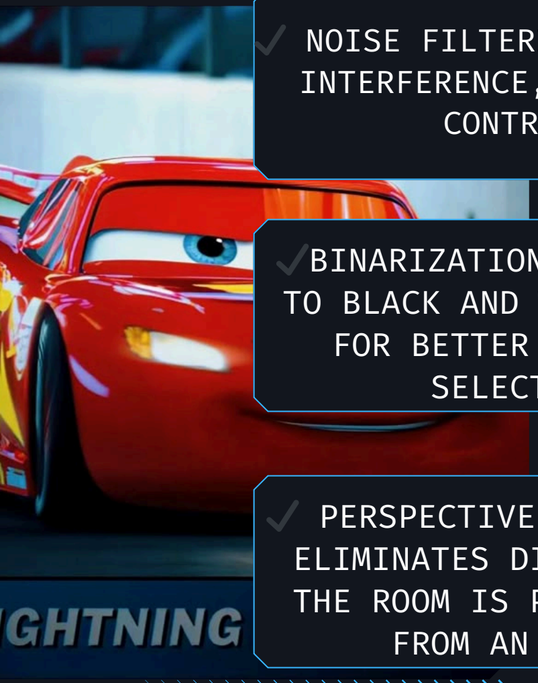
THE WORKING PROCESS OF THE MODEL

- **Loading the YOLOv8 model - The YOLOv8 nano is used, as it is lighter and faster on limited computing power. A separate model is loaded to detect license plates. license_plate_detector.pt .**
- **Vehicle Detection- Youloveit analyzes video stream frames and identifies objects related to vehicles. Data about the found objects (bounding boxes, class ID, confidence score) is saved for further processing.**
- **Detection filtering-Only cars, motorcycles, buses, and trucks are selected from all detected objects. The other classes of objects are ignored.**
- **License plate localization-A separate model is used to detect license plates (license_plate_detector.pt). The boundaries of the license plate on the vehicle are determined.**

This approach ensures efficient and accurate identification and tracking of vehicles, which is a key stage of the project.

IV TEXT RECOGNITION (OCR)

AFTER SUCCESSFULLY DETECTING LICENSE PLATES ON VEHICLES, THE SYSTEM USES OPTICAL CHARACTER RECOGNITION (OCR) TECHNOLOGY TO EXTRACT TEXTUAL INFORMATION. TEXT RECOGNITION (OCR) IS THE FINAL STAGE OF THE NUMBER PLATE RECOGNITION SYSTEM.

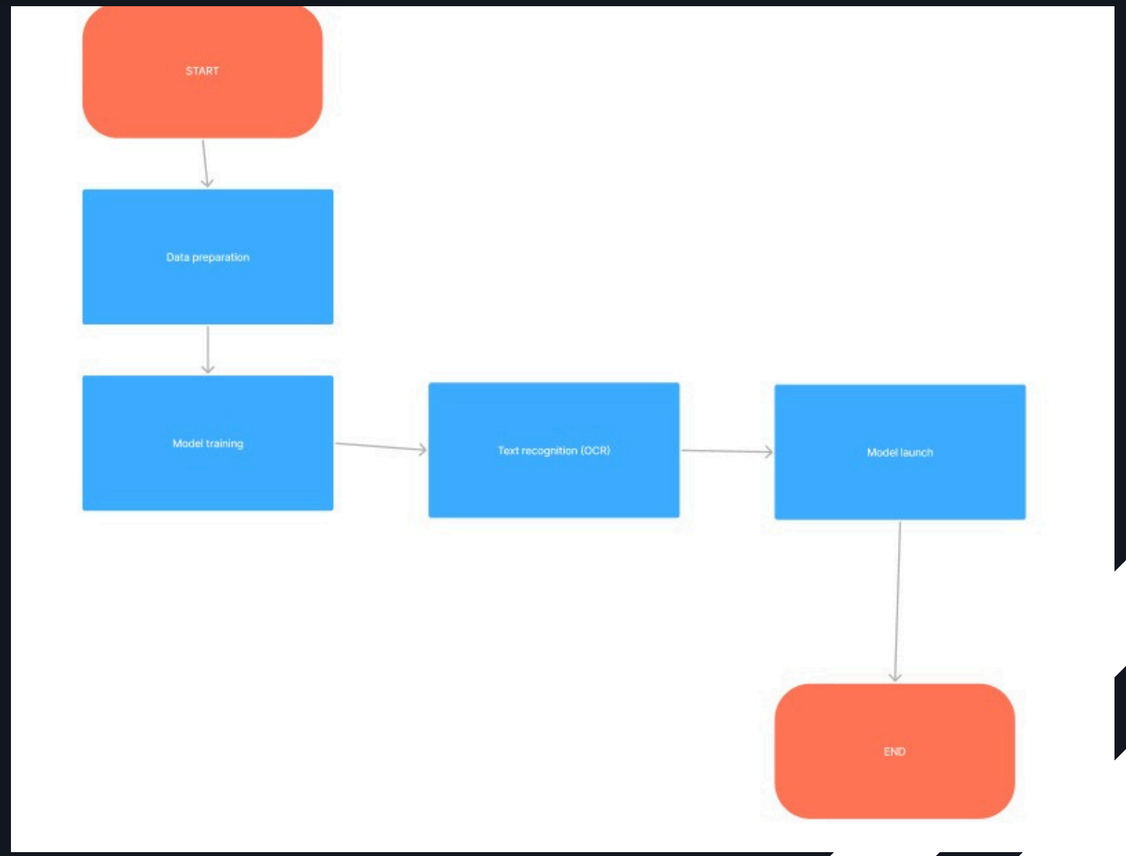


- NOISE FILTERING – REMOVING INTERFERENCE, INCREASING CONTRAST.
- BINARIZATION – CONVERSION TO BLACK AND WHITE FORMAT FOR BETTER CHARACTER SELECTION.
- PERSPECTIVE CORRECTION – ELIMINATES DISTORTION IF THE ROOM IS PHOTOGRAPHED FROM AN ANGLE.

SPECIALIZED MODELS SUCH AS TESSERACT OCR OR NEURAL NETWORK APPROACHES BASED ON CONVOLUTIONAL RECURRENT ARCHITECTURES ARE USED FOR RECOGNITION. TO INCREASE ACCURACY, THE NUMBER IMAGE IS PREPROCESSED BEFORE RECOGNITION.:

AFTER PROCESSING, OCR EXTRACTS TEXT CHARACTERS, WHICH ARE THEN CHECKED FOR CORRECTNESS BASED ON LICENSE PLATE PATTERNS. IF NECESSARY, POST-PROCESSING METHODS ARE USED, SUCH AS CORRECTING ERRORS USING LEXICAL RULES AND MACHINE LEARNING.THUS, THIS STAGE PROVIDES AUTOMATIC RECOGNITION OF STATE REGISTRATION PLATES, WHICH MAKES THE SYSTEM USEFUL FOR AUTOMATING TRAFFIC FLOW CONTROL, PARKING, AND ROAD MONITORING.

FLOW CHART


START
Data preparation
Model training
Text recognition (OCR)
Model launch
END

NUMBER PLATE RECOGNATION

Kalybek Aruzhan, Zhumabek Akbayan, Suleimenova Inkara

1. Background

Number Plate Recognition (NPR), also known as License Plate Recognition (LPR), is an advanced computer vision technology used to automatically detect and recognize vehicle license plates. Our project utilizes YOLO (You Only Look Once) for real-time object detection, enabling accurate identification of license plates in images and video streams. This approach, combined with machine learning and data processing tools like Roboflow, allows for efficient localization of plates under various conditions.

While traditional NPR systems rely on Optical Character Recognition (OCR) to extract alphanumeric characters, our focus is on enhancing detection accuracy and real-time performance through deep learning-based object detection. This technology is crucial for applications in traffic monitoring, security, and automation, reducing the need for manual supervision and improving efficiency.

2. Objectives

Aim
Our project focuses on developing an computer vision-powered Number Plate Recognition (NPR) system that can:

•Detect vehicles and license plates in images and video.
•Extract and display recognized plate numbers in real time.

Research Question
How accurately can deep learning models (YOLO) detect vehicles and license plates, and how effectively can extracted text be displayed for real-time applications?

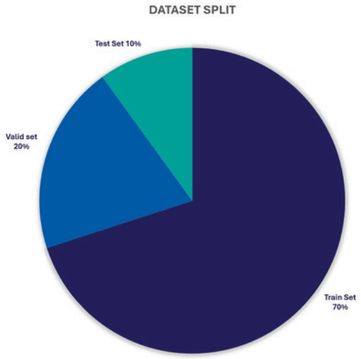
Objective
To create an automated, efficient, and scalable solution for vehicle recognition, enhancing applications in traffic management, security, and smart city systems.

3. Methods

Our project focuses on deep learning-based Number Plate Recognition (NPR) using YOLOv8 for object detection. The system was trained to identify vehicles and license plates in images and video streams. The dataset used for training comes from the Roboflow License Plate Recognition dataset, which contains annotated images of vehicles and plates captured under various conditions. Preprocessing steps included augmentation, resizing, and normalization to improve model performance.

The primary model for detection is YOLOv8 (You Only Look Once), while SVAT was initially considered but later deemed unnecessary. For text recognition, potential integration with OCR was explored to extract plate numbers. The system successfully detects vehicles and license plates in real time, displaying recognized plate numbers with high accuracy. Performance was further optimized through dataset tuning and model fine-tuning, ensuring reliable and efficient recognition.

3a. Dataset



The dataset provides the Train Set (7058 images), Valid Set (2048 images), Test Set(1020 images)

3b. Analysis

The License Plate Recognition dataset from Roboflow Universe provides annotated images designed for training machine learning models in automatic license plate detection and recognition. It includes a diverse range of vehicle license plates captured under different angles, lighting conditions, and environments. The dataset is available in multiple annotation formats, including COCO, VOC, and YOLO, making it compatible with various computer vision frameworks. This dataset is particularly useful for developing YOLOv8-based models for real-time vehicle monitoring, security applications, and smart parking systems.

4. Results

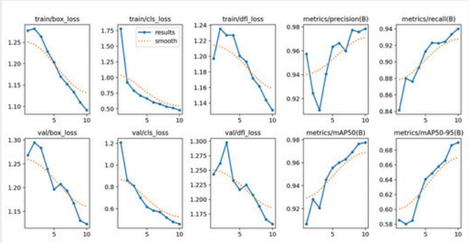


Fig 1. Graphs of YOLOv8 model learning metrics

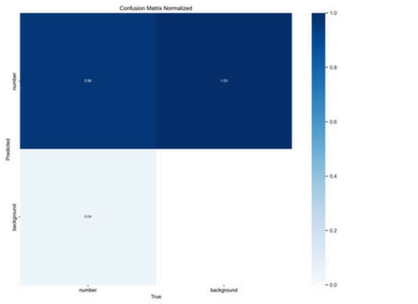


Fig 2. Confusion Matrix Normalized

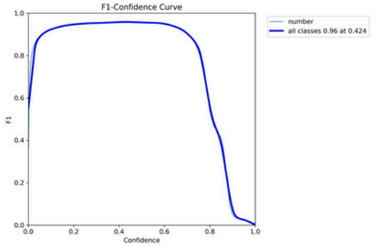


Fig 3. F1- Confidence Curve

Fig 4. Results

The model is learning steadily, and the metrics are improving (fig 1). However, the losses behave less smoothly on the validation data, which may indicate a slight instability or the initial stage of retraining. In general, the model works well, but it remains possible to reduce the 4% loss of numbers (fig 2). The F1-measure graph shows high values over a wide range of confidence, but a slight decrease is observed at high levels (Fig 3).Image analysis with number detection shows that the model successfully finds numbers in various conditions: on different types of cars (cars, buses, police cars), at night, at difficult viewing angles and in a busy environment. However, there are cases when the numbers are determined incorrectly, for example, in conditions of poor lighting or strong reflections. In some images, the model may have mistakenly marked other parts of the car or uncertainly identified the license plate, which may indicate the need for additional optimization (fig. 4).

5. Conclusion

The analysis showed that the developed model for number recognition demonstrates high quality indicators. During the training, there was a steady improvement in metrics, and the final results confirm the effectiveness of the chosen approach. The dataset used included images of cars in various conditions (different angles, lighting, types of cars), which allowed the model to successfully identify the numbers with high accuracy.

Nevertheless, the detected errors in detection on complex frames indicate possible areas of improvement. In particular, additional image processing techniques such as contrast enhancement or glare filtering can improve accuracy in night and illuminated scenes. In addition, balancing the training data and expanding the set of examples with atypical numbers can help the model better adapt to a variety of conditions.

In general, the goal of the work — to build a model capable of effectively recognizing car license plates in real conditions — has been achieved. However, further optimization will improve its stability and accuracy in complex scenarios.

References

Roboflow

Real-Time Automatic Number Plate Detection

Understanding Automated Number Plate Recognition Technology

CONCLUSION

THE DEVELOPED LICENSE PLATE RECOGNITION SYSTEM DEMONSTRATES EFFECTIVE VIDEO STREAM PROCESSING, INCLUDING VEHICLE DETECTION, LICENSE PLATE IDENTIFICATION AND OCR RECOGNITION. DURING THE PROJECT, KEY STAGES WERE IMPLEMENTED: DATA PREPARATION, MODEL SELECTION AND TRAINING, AND FINAL PROCESSING OF NUMBERS.SUCH A SYSTEM CAN BE USED FOR VARIOUS APPLICATIONS, INCLUDING PARKING ENTRANCE CONTROL, AUTOMATION OF FINES FOR TRAFFIC VIOLATIONS, AND TRAFFIC FLOW MONITORING. IN THE FUTURE, IT IS POSSIBLE TO IMPROVE RECOGNITION ACCURACY BY EXPANDING THE DATA SET, OPTIMIZING THE MODEL, AND USING ADDITIONAL IMAGE PREPROCESSING METHODS.THUS, THIS PROJECT IS A SIGNIFICANT STEP IN THE FIELD OF

THANK YOU

...