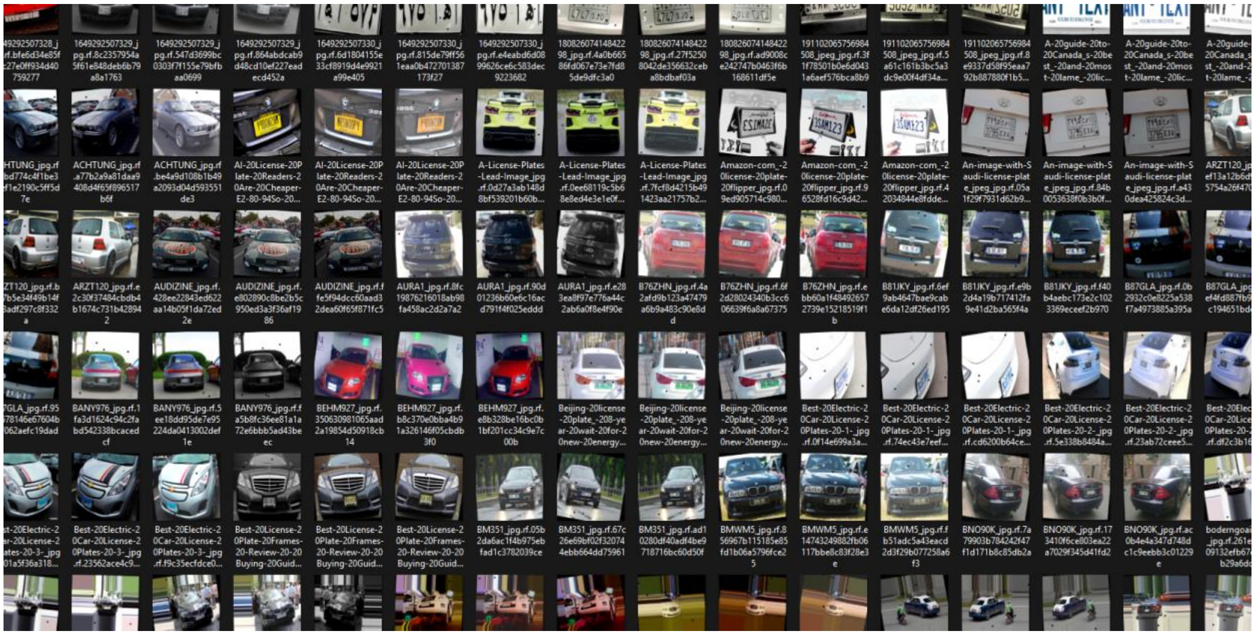
ANPR system for Uzbek car plates

System contains 2 parts, Plate detection from car images and OCR for recognition of numbers and letters from detected number plates.

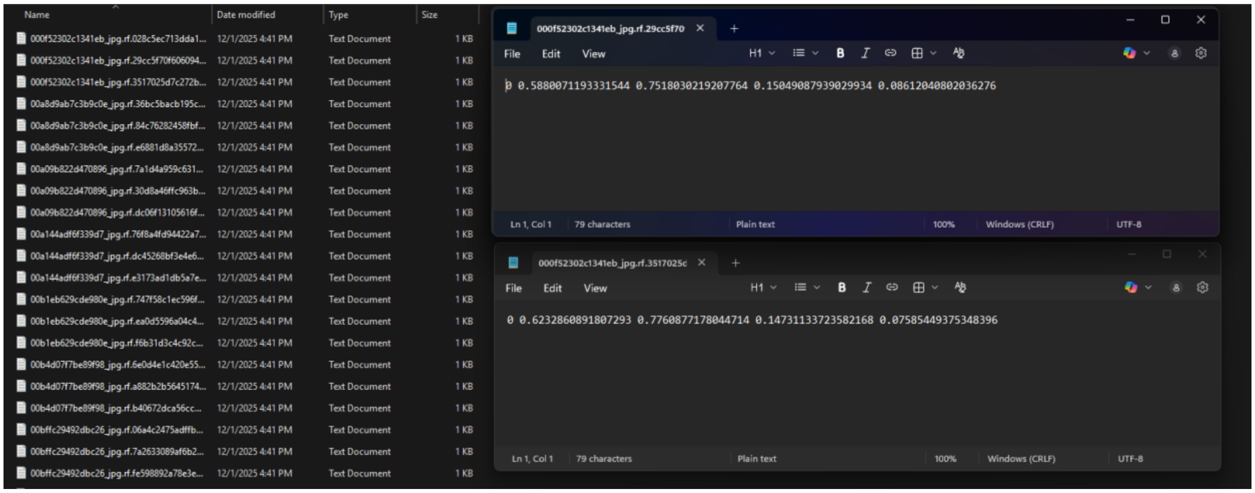
Models used: YOLO 11 for → number plate detection, LPRNet for → OCR system

Dataset:

1. Dataset for YOLO11 ---> 48k car images with number plates, found form Kaggle. 42k for training, 4k for validation and 2000 for testing. Img-1 shows example of data. Pictures multiplied by 3 , with rotating or changing lightness of an image. Img-2 is the example of annotations. Each txt file gives exact location of a number plate form an img, it works like coordinates.



Img-1



Name	Date modified	Type	Size
000f52302c1341eb_jpg.rf.028c5ec713dda1...	12/1/2025 4:41 PM	Text Document	1 KB
000f52302c1341eb_jpg.rf.29cc5f70f606094...	12/1/2025 4:41 PM	Text Document	1 KB
000f52302c1341eb_jpg.rf.3517025d7c272b...	12/1/2025 4:41 PM	Text Document	1 KB
00a8d9ab7c3b9c0e_jpg.rf.36bc5bacb195c...	12/1/2025 4:41 PM	Text Document	1 KB
00a8d9ab7c3b9c0e_jpg.rf.84c76282458fbf...	12/1/2025 4:41 PM	Text Document	1 KB
00a8d9ab7c3b9c0e_jpg.rf.e6881d8a35572...	12/1/2025 4:41 PM	Text Document	1 KB
00a09b822d470896_jpg.rf.7a1d4a959c631...	12/1/2025 4:41 PM	Text Document	1 KB
00a09b822d470896_jpg.rf.30d8a46ffc963b...	12/1/2025 4:41 PM	Text Document	1 KB
00a09b822d470896_jpg.rf.dc06f13105616f...	12/1/2025 4:41 PM	Text Document	1 KB
00a144adf6f339d7_jpg.rf.76f8a4fd94422a7...	12/1/2025 4:41 PM	Text Document	1 KB
00a144adf6f339d7_jpg.rf.dc45268bf3e4e6...	12/1/2025 4:41 PM	Text Document	1 KB
00a144adf6f339d7_jpg.rf.e3173ad1db5a7e...	12/1/2025 4:41 PM	Text Document	1 KB
00b1eb629cde980e_jpg.rf.747f58c1ec596f...	12/1/2025 4:41 PM	Text Document	1 KB
00b1eb629cde980e_jpg.rf.ea0d5596a04c4...	12/1/2025 4:41 PM	Text Document	1 KB
00b1eb629cde980e_jpg.rf.f6b31d3c4c92c...	12/1/2025 4:41 PM	Text Document	1 KB
00b4d07f7be89f98_jpg.rf.6e0d4e1c420e55...	12/1/2025 4:41 PM	Text Document	1 KB
00b4d07f7be89f98_jpg.rf.a882b2b5645174...	12/1/2025 4:41 PM	Text Document	1 KB
00b4d07f7be89f98_jpg.rf.b40672dca56cc...	12/1/2025 4:41 PM	Text Document	1 KB
00bffc29492dbc26_jpg.rf.06a4c2475adffb...	12/1/2025 4:41 PM	Text Document	1 KB
00bffc29492dbc26_jpg.rf.7a2633089af6b2...	12/1/2025 4:41 PM	Text Document	1 KB
00bffc29492dbc26_jpg.rf.fe598892a78e3e...	12/1/2025 4:41 PM	Text Document	1 KB

```
0 0.5880071193331544 0.7518030219207764 0.15049087939029934 0.08612040802036276
```

```
0 0.6232860891807293 0.7760877178044714 0.14731133723582168 0.07585449375348396
```

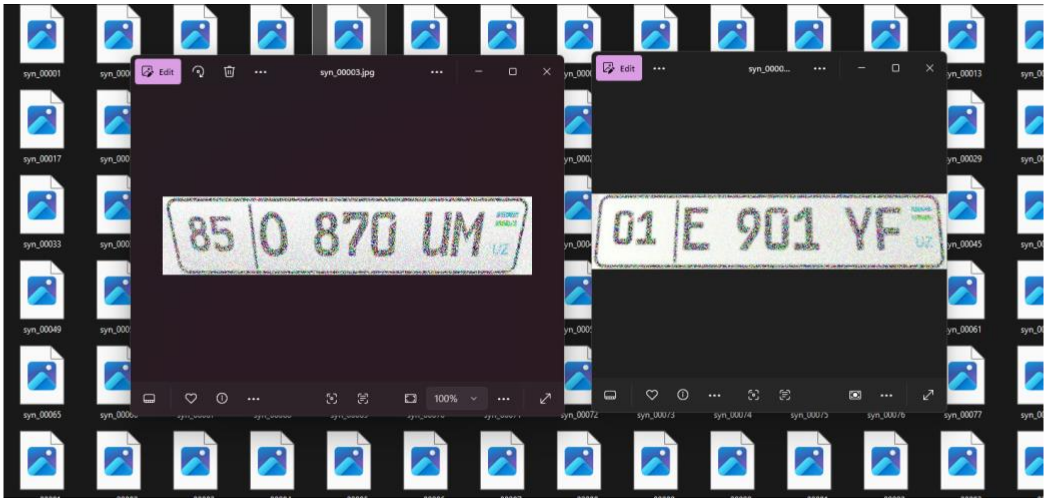
Img-2

2. Dataset for LPRNet ---> 1100 real Uzbekistan car plate cropped images and 5000 synthetic data created with pythin libraries

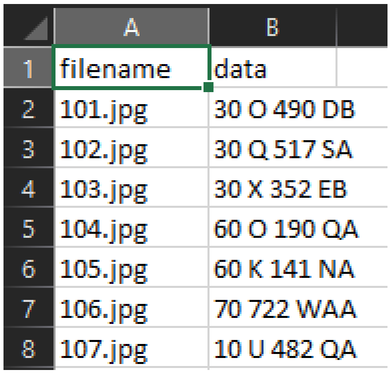
Img-3

Img-3 shows examples of real Uzbek car number plates. Downloaded from google images, and cropped from videos of dashcam. Moreover 5000 synthetic images created with python libraries, Image ImageDraw ImageFront. Img4 example of synthetic Uzbek number plate.

FE.TTF ---> font style is used for creating synthetic plates, FE font nearly same with uzbek plates. Img-5 shows data labelling format, 2 columns (filename and data)



Img-4



	A	B
1	filename	data
2	101.jpg	30 O 490 DB
3	102.jpg	30 Q 517 SA
4	103.jpg	30 X 352 EB
5	104.jpg	60 O 190 QA
6	105.jpg	60 K 141 NA
7	106.jpg	70 722 WAA
8	107.jpg	10 U 482 QA

Img-5

Moreover, Uzbekistan car number plates doesn’t continue letter “I”, so it have to be mentioned about it while training. And we wanted to train for private and company or organizational numbers, without special vehicles numbers. Example (01 A 111 AZ) (01 111 AAA). Next, region numbers are unique and they are: 01, 10, 20, 25, 30, 40, 50, 60, 70, 75, 89, 85, 90, 95. This keys alsa mentioned while tuning the models, cause then the model memorizes it and gives us significant change in accuracy.

Detection:

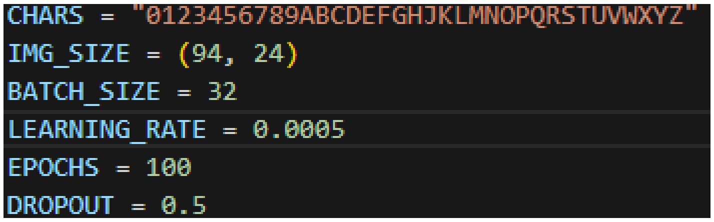
YOLO11 is fine tuned for number plate detection and **“yolo11s.pt”** was chosen. Epochs was 100, but in 24 it stopped training, cause model was already performing best via metrics. And now YOLO11s model is fine tuned with more thatn 40k images detects any kind a car plates with high precision nearly 99%.

Recognition - LPRNet

Choosing best ocr system for number plates was hard enough. Tested and tried to fine tune more than 10 OCR, PaddleOCR, EASYOCR, TesseractOCR, DocTR, FAST-ALPR and others. But stayed with LPRNet cause of its easy usage high for its high precion.

LPRNet gave the most highest accuracy among other OCRs, but still not high it is only 75% of accuracy which is not production ready. How I get 75? Tried this model with different kind of fine tuning version.

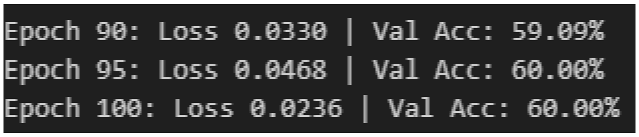
1st way,



```
CHARS = "0123456789ABCDEFGHJKLMNOPQRSTUVWXYZ"
IMG_SIZE = (94, 24)
BATCH_SIZE = 32
LEARNING_RATE = 0.0005
EPOCHS = 100
DROPOUT = 0.5
```

Img-6

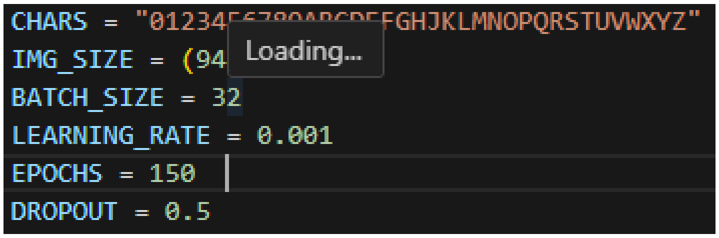
Moldel fine tuned with 1100 real manually collected and annotated dataset, and it gave this (Img-7):



```
Epoch 90: Loss 0.0330 | Val Acc: 59.09%
Epoch 95: Loss 0.0468 | Val Acc: 60.00%
Epoch 100: Loss 0.0236 | Val Acc: 60.00%
```

Img-7

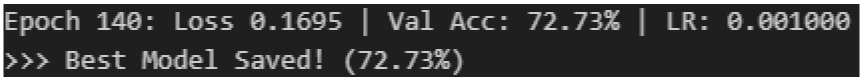
2nd way,



```
CHARS = "01234          FGHJKLMNOPQRSTUVWXYZ"
IMG_SIZE = (94
BATCH_SIZE = 32
LEARNING_RATE = 0.001
EPOCHS = 150
DROPOUT = 0.5
```

Img-8

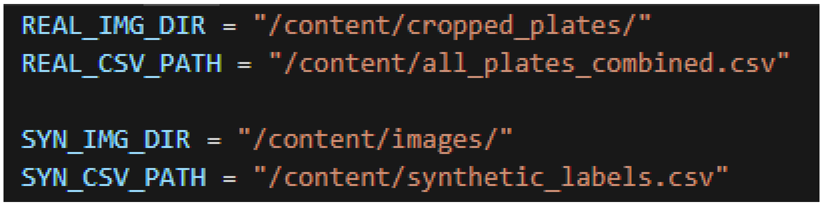
Metrics are changed and output is (Img-8):



```
Epoch 140: Loss 0.1695 | Val Acc: 72.73% | LR: 0.001000
>>> Best Model Saved! (72.73%)
```

3rd way,

Now tried to use real and synthetic data separately (Img-9)

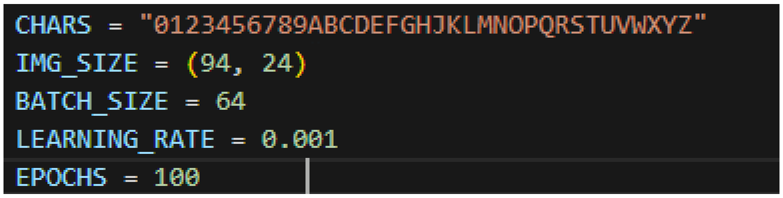


```
REAL_IMG_DIR = "/content/cropped_plates/"
REAL_CSV_PATH = "/content/all_plates_combined.csv"

SYN_IMG_DIR = "/content/images/"
SYN_CSV_PATH = "/content/synthetic_labels.csv"
```

Img-9

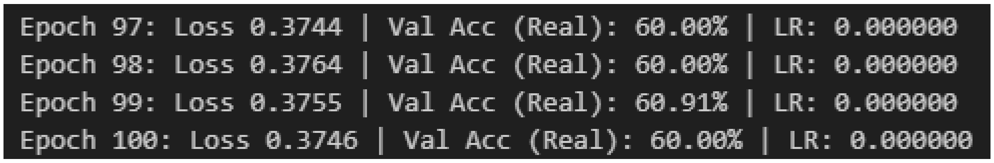
Metrics(Img-10) batch size increased and epochs decreased



```
CHARS = "0123456789ABCDEFGHJKLMNOPQRSTUVWXYZ"
IMG_SIZE = (94, 24)
BATCH_SIZE = 64
LEARNING_RATE = 0.001
EPOCHS = 100
```

Img-10

In 3rd try accuracy decreased from 72 to 60.9%

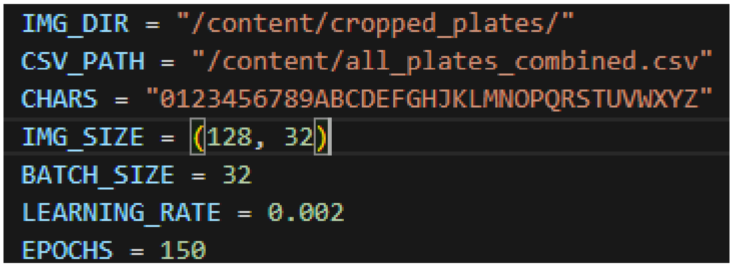


```
Epoch 97: Loss 0.3744 | Val Acc (Real): 60.00% | LR: 0.000000
Epoch 98: Loss 0.3764 | Val Acc (Real): 60.00% | LR: 0.000000
Epoch 99: Loss 0.3755 | Val Acc (Real): 60.91% | LR: 0.000000
Epoch 100: Loss 0.3746 | Val Acc (Real): 60.00% | LR: 0.000000
```

Img-11

4thway,

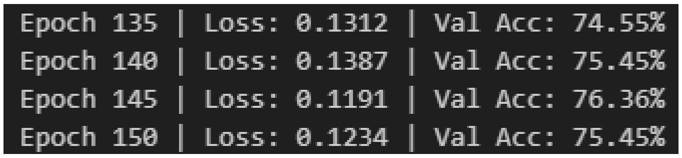
All data real and synthetic combined into one folder and annotation also put into 1 file



```
IMG_DIR = "/content/cropped_plates/"
CSV_PATH = "/content/all_plates_combined.csv"
CHARS = "0123456789ABCDEFGHJKLMNOPQRSTUVWXYZ"
IMG_SIZE = (128, 32)
BATCH_SIZE = 32
LEARNING_RATE = 0.002
EPOCHS = 150
```

Img-12

This changes played huge change in fine tuning and gave u 76.36% of accuracy which is our best till this .



```
Epoch 135 | Loss: 0.1312 | Val Acc: 74.55%
Epoch 140 | Loss: 0.1387 | Val Acc: 75.45%
Epoch 145 | Loss: 0.1191 | Val Acc: 76.36%
Epoch 150 | Loss: 0.1234 | Val Acc: 75.45%
```

Img-13

Results:

76.3% accuracy
PLAN: To increase it and get at least 90% , we need more at 2000 real Uzb car plate images to fine tune LPRNet and then it gives best production ready result. But Anyways 5000 img is perfect to fine tuning ocr model.

FINAL
ANPR system
Now, both models converted into ONNX for easy usage of models, and with FAST API our anpr system can done its duties. Latency is so low, that it will not take even a 1 second. Both of are working superfast in combination.

Thank you!

Jasurbek Usmonaliev

GITHUB:

https://github.com/Jasurbek701/ANPR-system