CAR PLATE RECOGNITION AND RECONSTRUCTION WITH DEEP LEARNING

Ciani Carlotta Anna Maria 1881291
Fuselli Michela 1883535
Laganà Simone Federico 1946083

Computer Vision 2024/25
Prof. Irene Amerini – June/July Exam Session 2025

OUTLINE

1 - THE PROBLEM

It is difficult to accurately recognize vehicle license plates in real-world conditions because of:

- partial occlusion,
- lighting variation,
- motion blur.

Traditional methods using:

- edge detector,
- color features analysis,
- morphological techniques for characters separation,

have limited performance in complex scenes.

We want to show that the new state of the art method is better than the traditional and simple methods.

2 - STATE OF THE ART

The problem of plate recostruction is usually solved in ***two separate steps***:

- **plate recognition**: where the plate region delimited by a ***bounding box*** is extracted from the image,
- **character recognition**: where the plate numbers and **characters** are recognized from the bounding box cropped image.

The state of the art methods of solving this problems involve **Deep learning** techniques, for both parts.

The best one, both in terms of execution time and overall accuracy, was the combination between **YOLOv5** for the plate recognition and **PDLPR** for the character recognition.

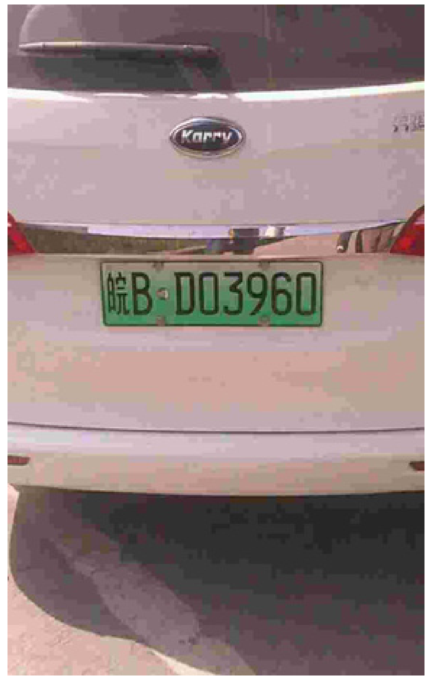
2 - YOLOv5

YOLO is a one-strage detection newtork that, given the image of a car, returns an image containing only the license plate

ARCHITECTURE

- **backbone**: is responsible for feature extraction from input images (using CSPDarknet)
- **neck**: makes it easier to aggregate features from different levels (using PANet to improve localization)
- **head**: generates the final prediction as output

INPUT



entire car image

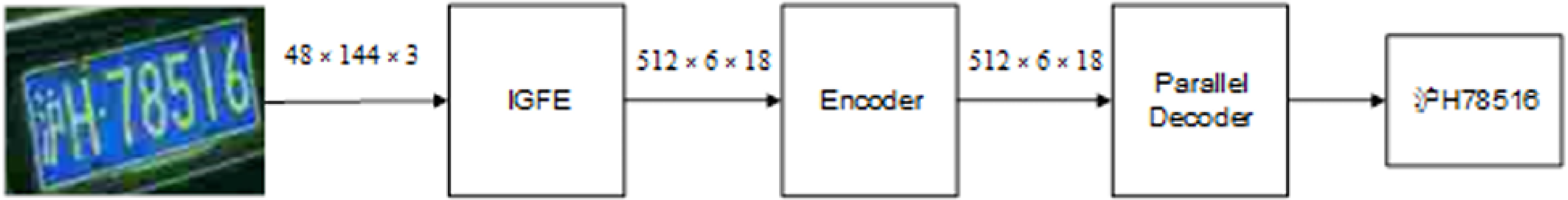
First, extract the coordinates of the bounding box, from the 3rd field of the filename
Second, crop images using the coordinates and create a .txt file containing the characters of the license plate.

OUTPUT

cropped plate image

2 - PDLPR

The PDLPR is the architecture used for the car plate OCR recognition.
It is composed by **three models**: IGFE, Ecnoder and Parallel decoder.



IGFE:

- FocusStructure → slicing + concat → conv
- ConvDownSampling ×2
- RESBLOCK ×4 → CNN blocks with residual connections

Encoder:

- CNN → MultiHeadAttention → CNN → Add&Norm
- positional encoding + multi-head attention

Parallel Decoder:

- MaskedMultiHeadAttention → MultiHeadAttention → FeedForward → Add&Norm

3 - PROPOSED METHOD

Our proposed baseline implementation uses both traditional computer vision techniques and deep learning and it is divided into two sections:

Traditional plate detection

Since the all the plates in the dataset have a green background and black characters the pipeline for this method is the following:

- Create a mask that isolates sections within a certain **green range**,
- Use **Canny edge detector** to extract the candidate boxes
- Compute the **IoU score** and using **OCR** to read the characters for each candidate box
- Select the **best box** according to the presence of text and IoU

3 - PROPOSED METHOD

Character recognition

It was chosen a **CNN neural network with CTC loss**. with this structure:

- 3 Convolutional layers + ReLU + MaxPool
- Permuting channels and width to treat width as **time steps**
- 2 Linear layers to reduce dimensionality
- output: n vectors (one per time step) each one with size the number of possible characters.

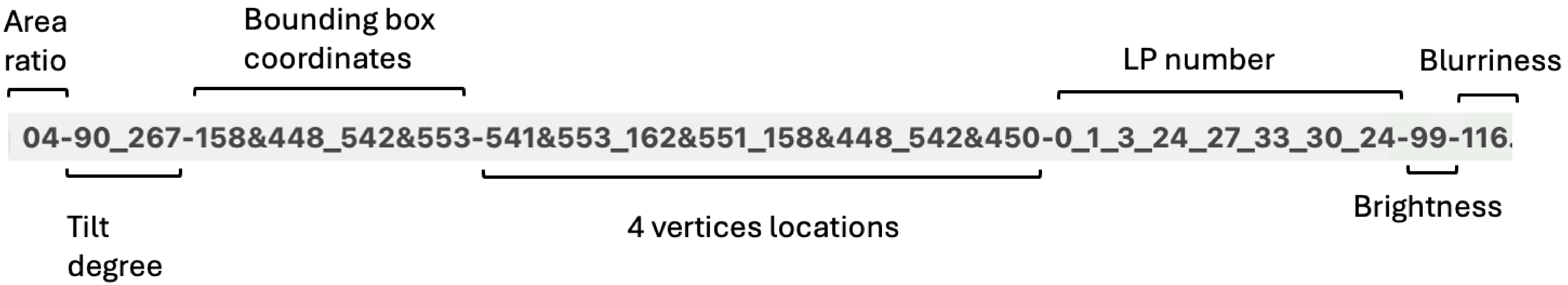
This model simulates a RNN, scanning the image left to right across its reduced width which can be interpreted as a sequence of time steps.
It outputs a series of **prediction logits** which are decoded into the final character predictions.

4 - DATASET

CCPD (Chinese City Parking Dataset, ECCV) - UPdate on 16/09/2020

- over 250k annotated images
- one license plate per image
- **8 characters** per license plate
- 7 fields in the image name

Image name structure



5 - EXPERIMENTAL SETUP

Framework: PyTorch, trained on local CPU, MPS, and GPU on Google Colab

Hyperparameters: Learning rate, weight decay, num_epochs, batch_size.

YOLOv5	Input: 640x640 resized images Output: bounding box coordinates in this format [x1, y1, x2, y2]
PDLPR	Input: x -> [B, 3, 48, 144] RGB image tensor, resized Output: features -> [B, 512, 6, 18] 512-channel feature map with spatial structure
Traditional plate recognition	Input: raw image Output: bounding box coordinates [x1, y1, x2, y2]
CNN + CTC	Input: x -> [B, 1, 48, 144] Grayscaled image tensor, resized Output: features -> [36, B , Num_characters] logits
Loss Function	CTC Loss (blank token = 0)
Vocab size	40 (Chinese provinces + letters + digits)

6 - MODEL EVALUATION

The models were evaluated using the following metrics:

For the **plate detection:**
- **YOLOv5** → **Intersection over Union** with threshold **0.7**
- **Traditional** plate detector → **Intersection over Union** together with **OCR** flag (if there are any characters presents), testing different combinations for the green tone

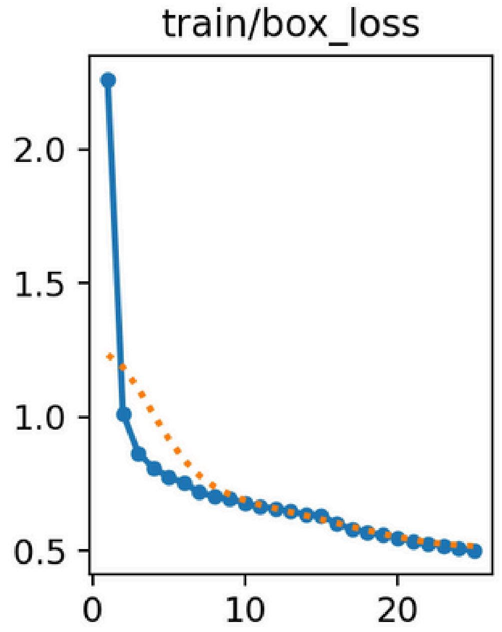
For **character recognition**:
- PDLPR & CNN CTC → **Sequence accuracy** & **character accuracy**
  - the **first one** describes the accuracy in predicting the full plate,
  - the latter the amount of correct characters that are guessed

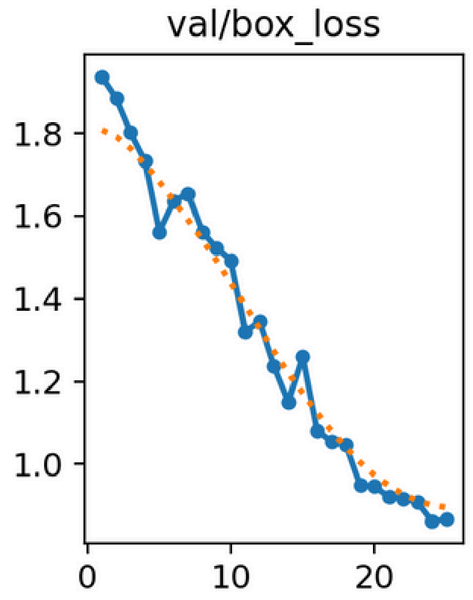
Pipelines
We evaluated the pipeline execution of the two parts in order to get the sense of a real life application, where the output of the plate detection part is directly passed to the character recognition part and then evaluated.

EVALUATION - YOLOV5

RESULTS - YOLO - TRAIN

Number of epochs	Batch size	Learning rate	IoU
20	8	0,001	0,8553
20	20	0,002	0,8679
25	20	0,001	**0,8737**
30	12	0,001	0,8630





EVALUATION - YOLOV5

RESULTS - YOLO - TEST

Number of epochs	Batch size	Learning rate	mean IoU
20	8	0,001	0,8759
20	20	0,002	0,8858
25	20	0,001	**0,8924**
30	12	0,001	0,8846

EVALUATION - PDLPR

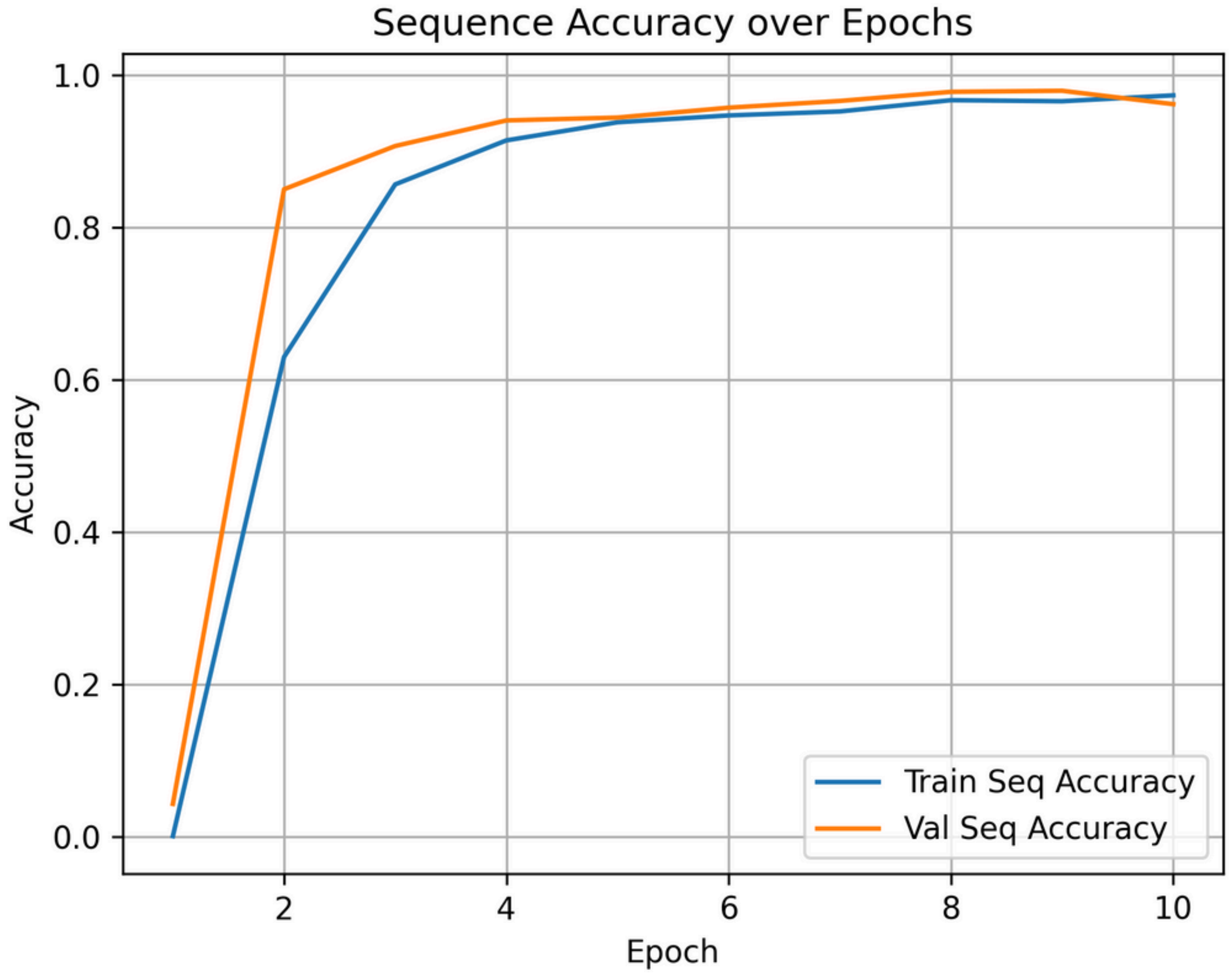
RESULTS - PDLPR TRAIN

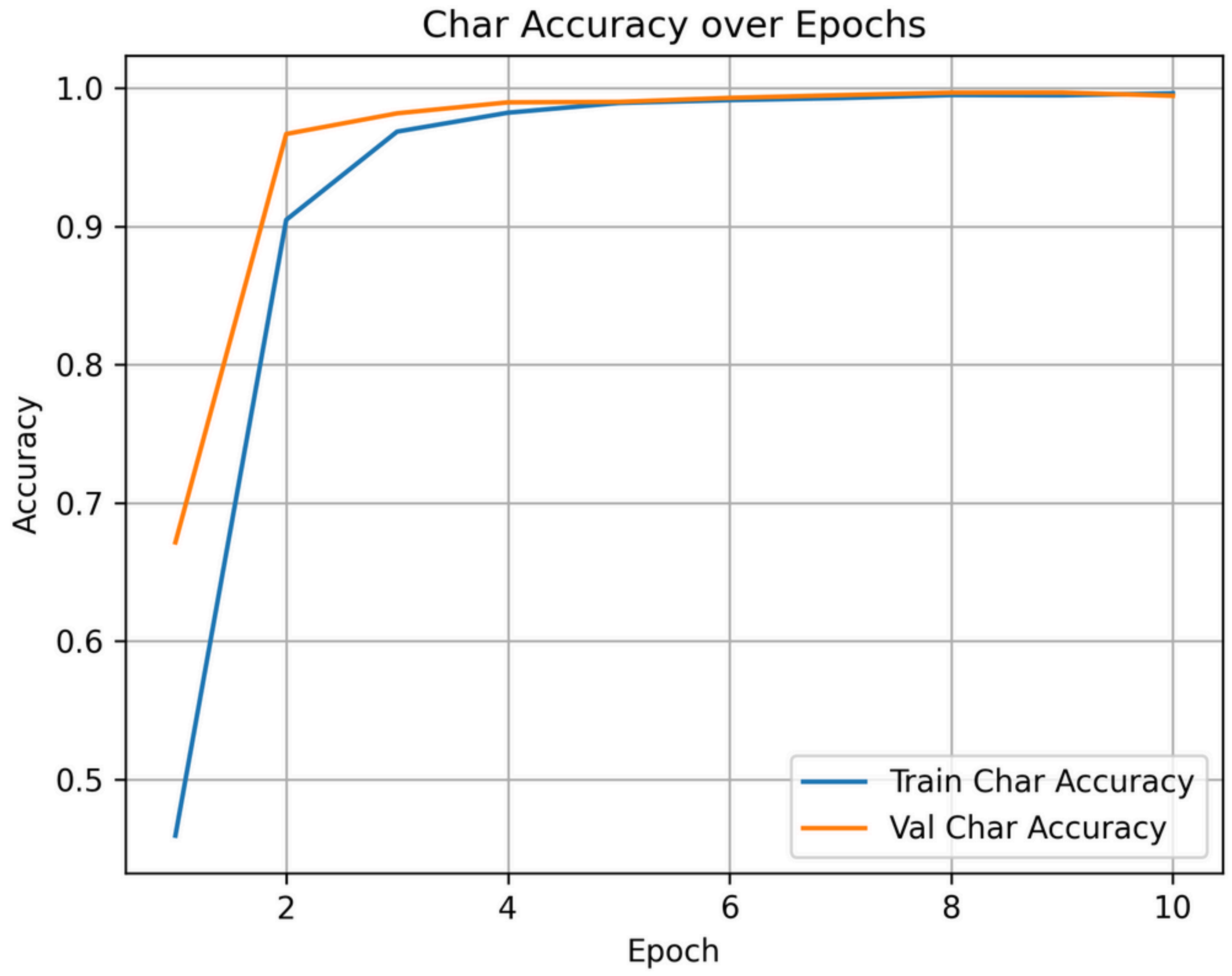
N epochs	Batch size	Learning rate	Train seq_ accuracy	Train char accuracy	Val seq accuracy	Val char accuracy
5	16	0.00001	0.9365	0.9886	0.9469	0.9921
10	**16**	**0.00001**	**0.9732**	**0.9962**	**0.9616**	**0.9962**
5	32	0.00001	0.9137	0.9829	0.9303	0.9829

EVALUATION - PDLPR

RESULTS - PDLPR TRAIN

Epochs: 10, learning rate: 0.00001, batch size 16 - best model accuracy plots





EVALUATION - PDLPR

RESULTS - PDLPR TESTING

Number of epochs	Batch size	Learning rate	Test seq accuracy	Test char accuracy
10	**16**	**0.00001**	**0.9616**	**0.9944**
5	16	0.00001	0.9469	0.9921
5	32	0.00001	0.9303	0.9884

EVALUATION - TRAD. PLATE DETECTION

RESULTS

Lower green	Upper green	Average IoU	IoU pass rate
[35, 40, 40]	[85, 255, 255]	0,3840	27,20
[45, 80, 60]	[75, 255, 255]	0,4365	24,49
[40, 50, 50]	**[85, 255, 255]**	**0,4752**	**36,89**
[40, 40, 40]	[80, 255, 255]	0,4362	31,53

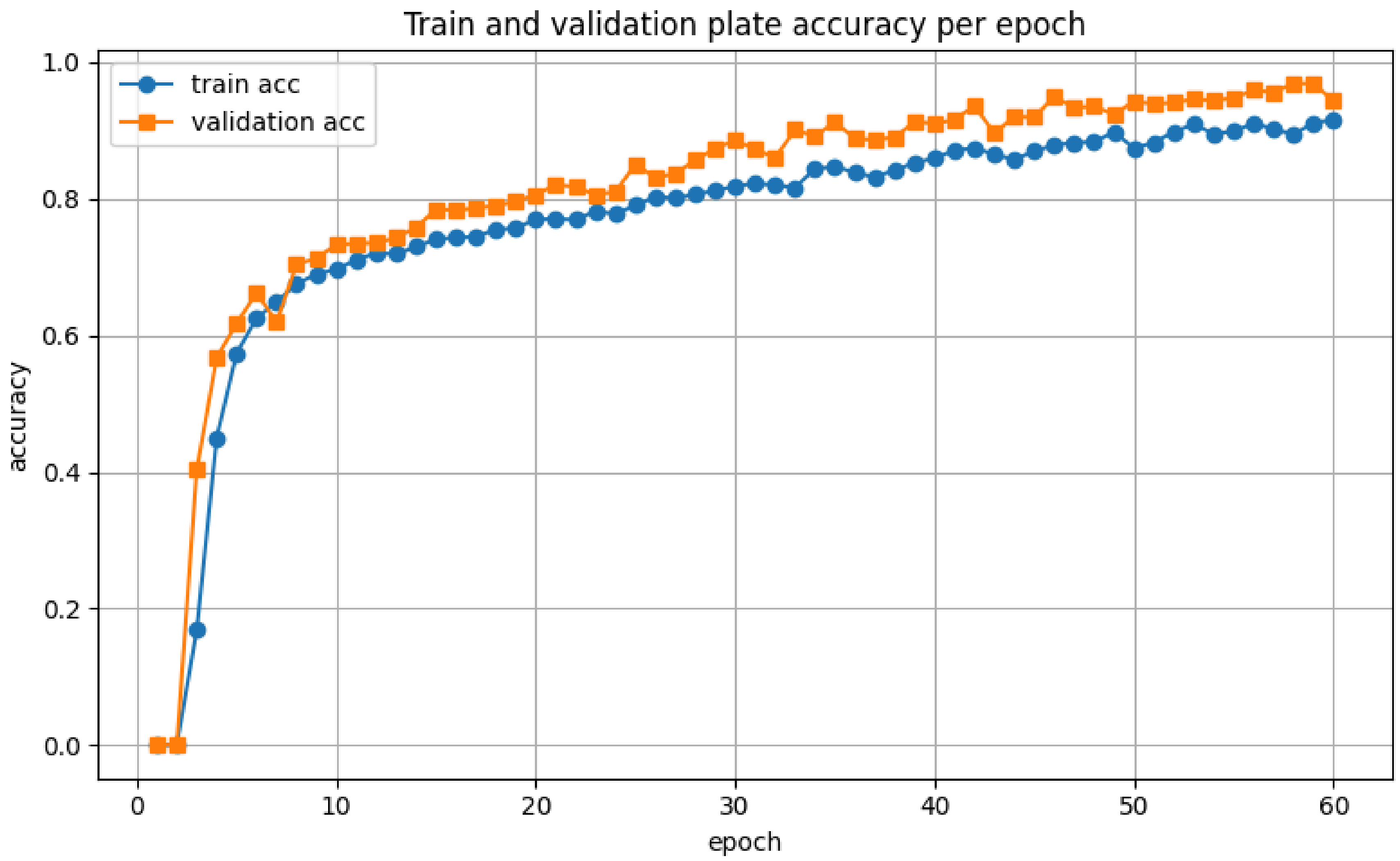
EVALUATION - CNN CTC

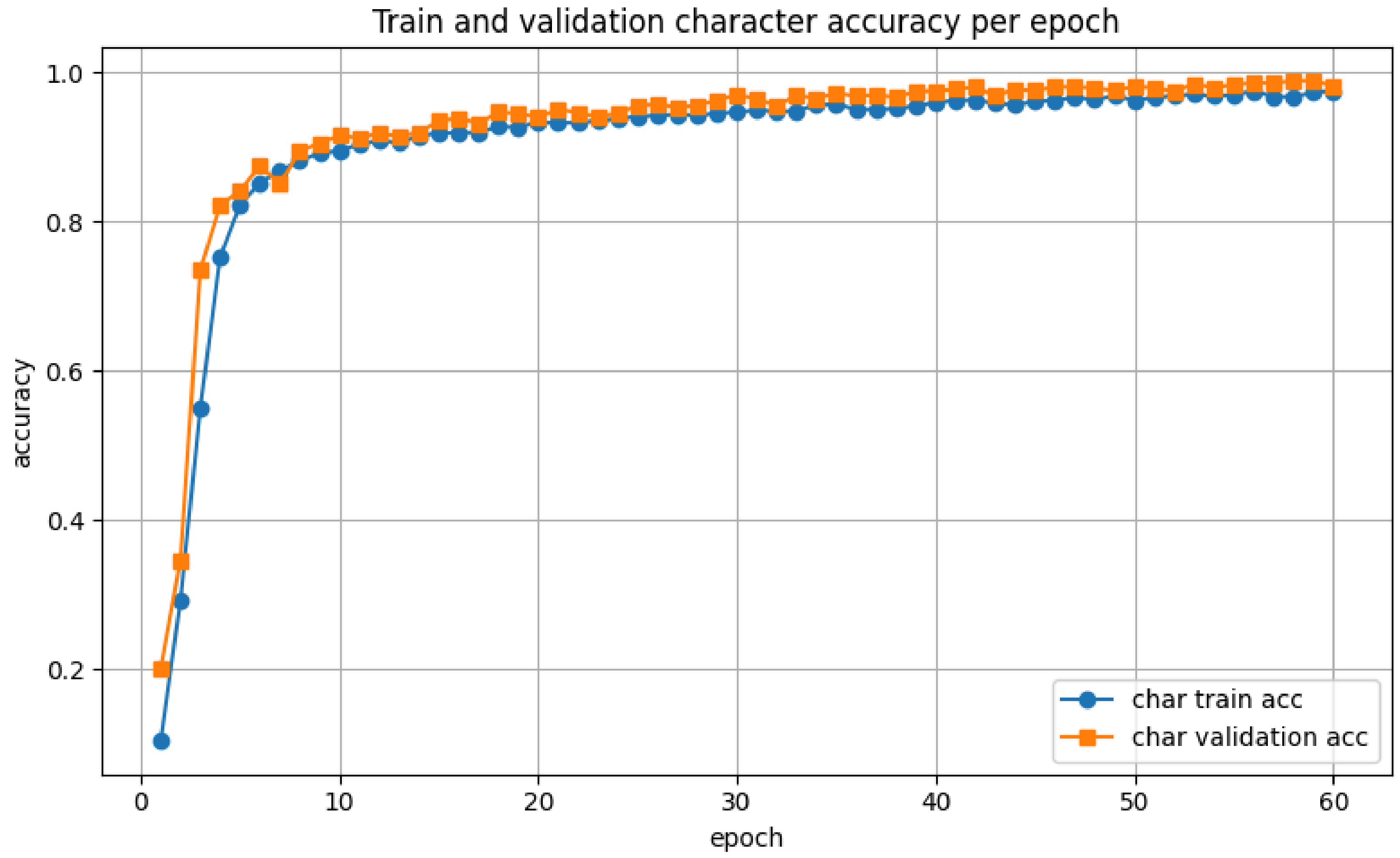
RESULTS - CNN CTC TRAIN

N epochs	Batch size	Learning rate	Weight decay	Train seq_ accuracy	Train char accuracy	Val seq accuracy	Val char accuracy
60	**32**	**0.001**	**0.0001**	**0.9159**	**0.9759**	**0.9439**	**0.9825**
40	64	0.001	0.0005	0.7527	0.9263	0.7933	0.9431
40	64	0.001	0.0001	0.7762	0.9356	0.8083	0.9382
40	32	0.001	0.0001	0.8471	0.9572	0.9037	0.9721

EVALUATION - CNN CTC

RESULTS - CNN CTC TRAIN BEST





EVALUATION - CNN CTC

RESULTS - CNN CTC TEST

Number of epochs	Learning rate	Batch size	Weight decay	seq accuracy	char accuracy
60	**0.001**	**32**	**0.0001**	**0.9439**	**0.9825**
40	64	0.001	0.0005	0.7933	0.9431
40	64	0.001	0.0001	0.8083	0.9382
40	32	0.001	0.0001	0.9037	0.9721

EVALUATION - PIPELINES

RESULTS

These results were computed among the test dataset, taking one image at a time.

PIPELINE	mean seq accuracy	mean char accuracy	mean iou score
YOLO + PDLPR	**0.8561**	**0.9702**	**0.8737**
TRAD. DETECTION + CNN CTC	0.0408	0.3932	0.4166

7 - CONCLUSIONS

We concluded that the state-of-the-art pipeline of **Yolov5 and PDLPR** had **better performances** under all the metrics, both in individual evaluation and in the pipeline evaluation. Especially in plate recognition, since yolo is better at handling different contexts.
We can also see that it takes much less epochs to train respect to the proposed approach.

Future work
As future additions there is the one of creating models that could detect **multiple kinds of plates** not only cars, e.g. motorcicles, trucks etc.. also in **different countries**.

To create more flexible models it could be implemented **data augmentation** by creating new images that are mirrored, with low contrast, distorted etc..

THANK YOU FOR YOUR ATTENTION

References

Tao, L.; Hong, S.; Lin, Y.; Chen, Y.; He, P.; Tie, Z. A Real-Time License Plate Detection and Recognition Model in Unconstrained Scenarios. Sensors 2024, 24, 2791. https://doi.org/10.3390/s24092791

https://docs.ultralytics.com/it/models/yolov5/

https://github.com/detectRecog/CCPD

https://www.mindspore.cn/tutorials/application/en/r2.2/cv/cnnctc.html