License Plate Detection and Recognition

Comparative Study between Baseline and Yolov5-PDLPR

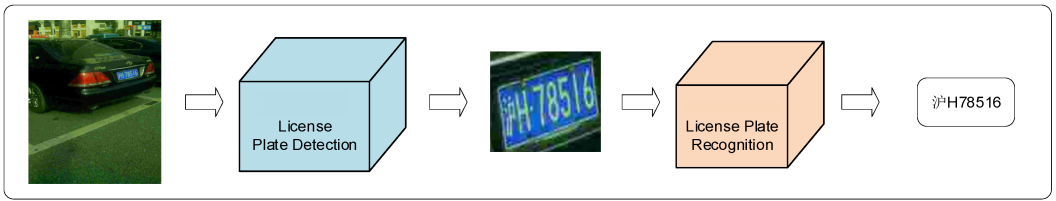
Master's Degree in Artificial Intelligence and Robotics

Computer Vision

Martina Giusti, Francesco Tomaselli

Academic Year 2024/2025

SAPIENZA
UNIVERSITÀ DI ROMA



The project focuses on the development of an advanced system for automatic license plate detection and recognition using deep learning techniques.

After implementing a baseline, the YOLOv5-PDLPR model for license plate detection and recognition were reproduced.

Table of Contents

1 Dataset e Setup Hardware

Overview of the CCPD Dataset (Chinese City Parking Dataset)

- Over **250.000** unique license plate images;
- **Resolution:** 720 × 1160 × 3;
- **Average size per image:** 200KB;
- **Dataset size:** 48 GB

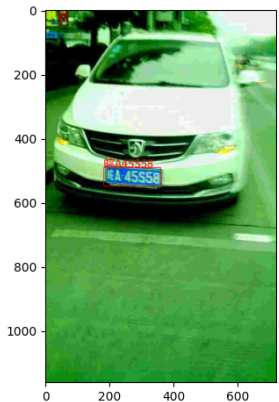
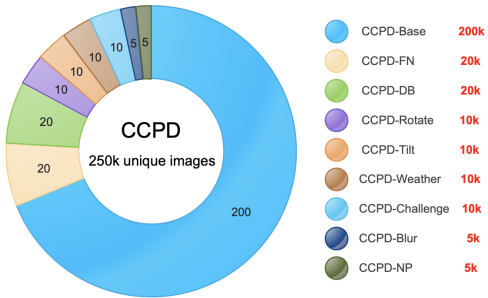


Image Division in CCPD

1 Dataset e Setup Hardware



- **CCPD-Base:** Split into **train** and **validation**;
- **CCPD-DB** (Lighting variations), **CCPD-Blur** (Blurred images), **CCPD-FN** (Variable distance of the plate), **CCPD-Rotate** (Rotated plates), **CCPD-Tilt** (Tilted plates), **CCPD-Challenge** (Particularly difficult images): for **test**.

Annotations in CCPD

Among all the annotations present in the dataset, the following were used for the project:

- **License Plate (label)**: Composed of one Chinese character, one letter, and five alphanumeric characters;
- **Bounding Box**: Coordinates (x, y) of the top-left and bottom-right corners of the license plate.

Hardware Used

1 Dataset e Setup Hardware

TensorDock

The model training was carried out on a remote server provided by **TensorDock**, with the following specifications:

- **GPU:** 1× NVIDIA RTX 4000 ADA (20 GB VRAM);
- **CPU:** 8× vCPU;
- **RAM:** 28 GB;
- **Storage:** 250 GB NVMe SSD.

Table of Contents

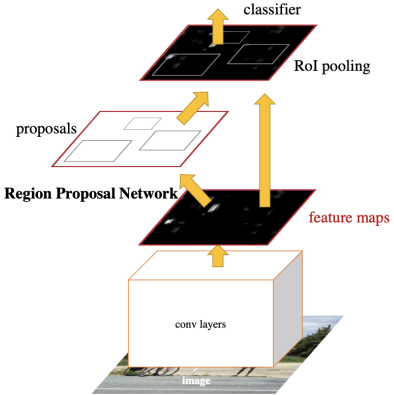
2 Detection: Baseline

Faster R-CNN: Fine-Tuning on CCPD

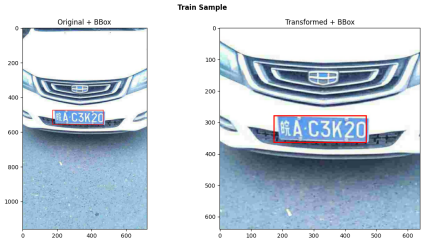
2 Detection: Baseline

The **fine-tuning** of **Faster R-CNN** was performed on the CCPD dataset.

- **Two-stage architecture**;
- **Unified neural network**, integrating region generation and classification in a single shared architecture;
- **Higher accuracy**, but **slower inference** compared to one-stage models;
- Excellent as a **baseline** for comparisons with YOLOv5.



Preprocessing for Faster R-CNN



Preprocessing was applied to the CCPD images for consistency and direct comparison with YOLOv5.

- **Centered cropping** on the license plate, fixed size (640×640);
- **Updating of bounding box coordinates**;
- **Boxes in absolute coordinates**.

Augmentation for Faster R-CNN

2 Detection: Baseline

Simple **data augmentation** techniques with configurable probability were applied:

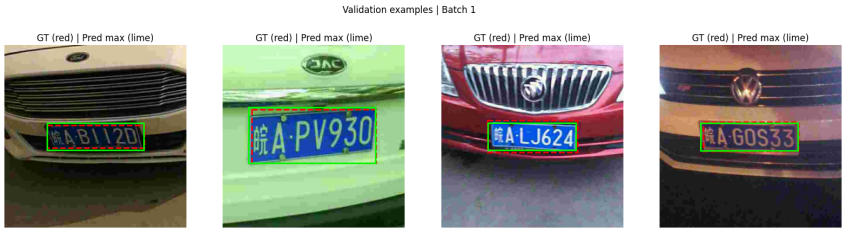
- **Random horizontal flip** with updated coordinates;
- **Color jitter**: variation of brightness, contrast, saturation, and hue;
- **Additive Gaussian noise** with zero mean and controlled standard deviation.

Training Faster R-CNN

- **Batch size**: 4
- **Epochs**: 15 (3 hours per epoch)
- **Model**: fasterrcnn_resnet50_fpn (pre-trained)
- **Optimizer**: SGD (momentum 0.9)
- **Initial learning rate**: $5 \cdot 10^{-3}$
- **Scheduler**: StepLR reduction (every 3 epochs)

Validation Faster R-CNN

2 Detection: Baseline



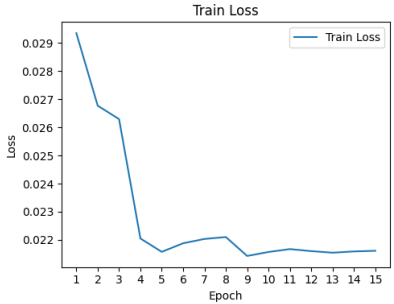
- **Evaluation using IoU**:

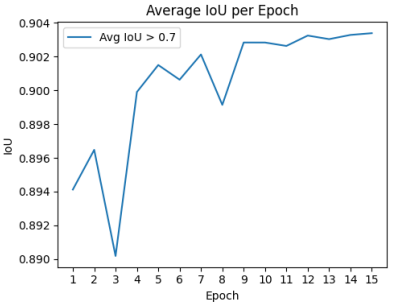
$$\text{IoU} = \frac{\text{Area}(db \cap gb)}{\text{Area}(db \cup gb)} > 0.7$$

- **Metric used**: Average IoU calculated over all images in the validation set.

Training and Validation Results

2 Detection: Baseline





- **Epoch:** 15
- **Train Loss:** 0.0216
- **Avg IoU** > 0.7 **(validation):** 0.9034

Test Results and Comparison with Other Methods (CCPD)

2 Detection: Baseline

Method	Overall Accuracy	Base (100k)	DB (20k)	FN (20k)	Rotate (10k)	Tilt (10k)	Weather (10k)	Challenge (10k)	Blur (20k)	Speed (FPS)
Faster RCNN	92.9	98.1	92.1	83.7	91.8	89.4	81.8	83.9	-	17.6
YOLO9000	93.1	98.8	89.6	77.3	93.3	91.8	84.2	88.6	-	43.9
SSD300	94.4	99.1	89.2	84.7	95.6	94.9	83.4	**93.1**	-	40.7
TE2E	94.2	98.5	91.7	83.8	95.1	94.5	83.6	**93.1**	-	3.2
RPnet	94.5	**99.3**	89.5	85.3	94.7	93.2	84.1	92.8	-	85.5
YOLOv5	**96.7**	97.2	**97.7**	**92.9**	**98.9**	**98.9**	**99.0**	90.6	-	**218.3**
Ours F-RCNN	85.03	90.34	82.97	82.38	84.47	80.64	89.27	85.11	80.3	27.26

Comparison of performance on CCPD test subsets.
The best result is in bold, and the second best is underlined.

Table of Contents

Deep Text Recognition Benchmark

3 Recognition: Baseline

- **Deep Text Recognition Benchmark**: framework for **Scene Text Recognition** (STR), i.e., the recognition of text in natural scenes.
- **4 main stages** of the STR model:
  1. **Transformation**: image normalization;
  2. **Feature Extraction**;
  3. **Sequence Modelling**;
  4. **Prediction**.
- **Modularity**: allows testing of different combinations of modules across the 4 stages.
- **Pre-trained models**: fine-tuned for the specific case study of License Plate Recognition.
- The possibility of introducing other characters and alphabets.

Deep Text Recognition Benchmark: STAR-Net

3 Recognition: Baseline

Referring to the work by [1], **STAR-Net**, the most effective combination in terms of accuracy, inference time per image, and number of parameters with respect the other modules combination, was selected:

1. **TPS** (Thin-Plate Spline) for *Transformation*;
2. **ResNet** for *Feature Extraction*;
3. **BiLSTM** for *Sequence Modelling*;
4. **Attn** (Attention) for *Prediction*.

This structure was implemented. The dataset was created following the instructions of the GitHub repository [1], starting from the license plates images, cropped around the bounding box and resized to a 32×100 resolution image.

The vocabulary problem

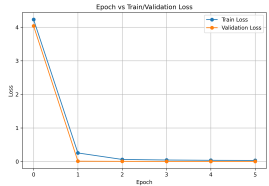
The pre-trained model was taken into account with its weights.
The STAR-Net pre-trained model is able only to recognize alphanumeric characters.
To overcome this problem, the character vocabulary was expanded by adding the Chinese characters for the province, leading to adding other weights because the alphabet is larger than before. The vocabulary was modified in order to decode only the characters of our case of study:

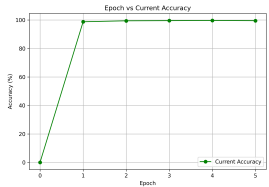
0123456789ABCDEFGHIJKLMNPQRSTUVWXYZ
皖沪津渝冀晋蒙辽吉黑苏浙京闽赣鲁豫鄂湘粤桂琼川贵云藏陕甘青宁新警学O

STAR-Net: Training settings

3 Recognition: Baseline

- Batch size : 48;
- Image resize to 32×100, by default;
- Number of epochs: $5 \rightarrow 10\,420$ iterations
  1 epoch $= \frac{\text{\# train elements}}{\text{batch size}} = \frac{100\,000}{48} = 2\,084$ iterations;
- Number of input channels: 1;
- Number of output channels: 512;
- Word maximum length: 7 characters;
- Augmentation is applied.





Augmentation

The original code did not provide any augmentation; therefore, augmentation was introduced in the training pipeline via the function *AlignCollate* in *deep-text-recognition-benchmark/dataset.py*.

Applied Augmentation Techniques:

- **Random rotation:**
  $\pm 3^\circ$
- **Random affine transformation:**
  - Rotation: $\pm 5^\circ$;
  - Translation: $\pm 2\%$ $(0.02, 0.02)$;
- **Color jitter:**
  - Brightness: $\pm 20\%$;
  - Contrast: $\pm 20\%$;
- **Gaussian blur:**
  - Kernel size: 3×3;
  - Sigma: $0.1 - 2.0$;

Some example of augmented plates

Figure: Example of augmented plates taken from the train set

Performance on test sets

3 Recognition: Baseline

Folder	Number of samples	Time of inference (s)	Accuracy	FPS
Blur	20611	382.645	98.43%	53.86
Challenge	50003	1076.77	97.97%	46.44
Dark/Bright (db)	10132	262.924	96.69%	38.54
Far/Near (fn)	20967	333.647	98.15%	62.84
Rotate	10053	189.914	99.16%	52.93
Tilt	30216	468.838	98.47%	64.45
Weather	9999	151.3	99.46%	66.09

Test Results and Comparison with Other Methods (STAR-Net)

3 Recognition: Baseline

Method	Overall Accuracy	Base (100k)	DB (20k)	FN (20k)	Rotate (10k)	Tilt (10k)	Weather (10k)	Challenge (10k)	Blur	Speed (FPS)
Zou et al.	97.8	99.3	98.5	98.6	92.5	96.4	99.3	86.6	-	-
Zhang et al.	98.5	99.6	98.8	98.8	96.4	97.6	98.5	88.9	-	40.2
Zhang et al.	98.9	99.8	99.2	99.1	98.1	98.8	98.6	89.7	-	40.2
Qin et al.	97.2	99.3	92.9	93.2	97.9	95.5	98.8	92.4	-	36.0
Qin et al.	97.6	99.5	93.3	93.7	98.2	95.9	98.9	92.9	-	26.0
Fan et al.	98.8	99.7	99.1	99.0	99.1	99.0	98.5	88.0	-	11.7
Fan et al.	99.0	99.8	99.2	99.2	**99.6**	**99.6**	98.5	88.8	-	26.0
PDLPR	**99.4**	**99.9**	**99.5**	**99.5**	99.5	99.4	99.4	94.1	-	**159.8**
Ours STAR-Net	98.5	99.6	96.69	98.15	99.16	98.47	**99.46**	**97.97**	98.43	55.02

Comparison of performance on CCPD test subsets. Best results in bold, second best in underlined.

Table of Contents

4 Detection: YOLOv5

YOLOv5: Fine-Tuning on CCPD

A **fine-tuning** of the `yolov5s` model was performed on the **CCPD** dataset.

Main Features of YOLOv5s:

- **Input size:** 640×640 pixels;
- **mAP@0.5:** 56.8% (COCO);
- **Parameters:** 7.2M;
- **Real-time capable**: excellent trade-off between speed and accuracy.

Creation of the Dataset for YOLOv5

A YOLO formatted dataset was created for training and evaluation of the model, following these main steps:

- **Transformation of images**;
- **Saving the images**;
- **Saving the labels** in YOLO format (class_id, x_center, y_center, width, height)

Creation of the Dataset for YOLOv5

YOLOv5 Folder Structure

```
YOLO_format/
├── images/
│   ├── train/
│   ├── val/
│   └── test/
├── labels/
│   ├── train/
│   ├── val/
│   └── test/
```

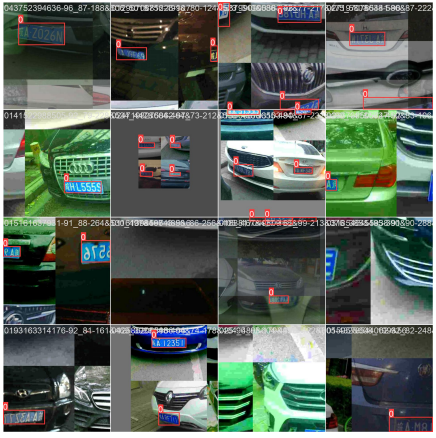
Number of Images per Split

- **Train Split:** 100.000 images
- **Validation Split:** 99.996 images
- **Test Split:** 100.579 images

Training

4 Detection: YOLOv5

- **Batch size**: 50
- **Epochs**: 30 (vs 300 in the paper)
- **Initial weights**: yolov5s.pt
- **Optimizer**: Adam
- **Initial Learning Rate**: $1 \cdot 10^{-3}$
- **Final Learning Rate**: $1 \cdot 10^{-5}$



Example of a training batch with **labels**

Validation

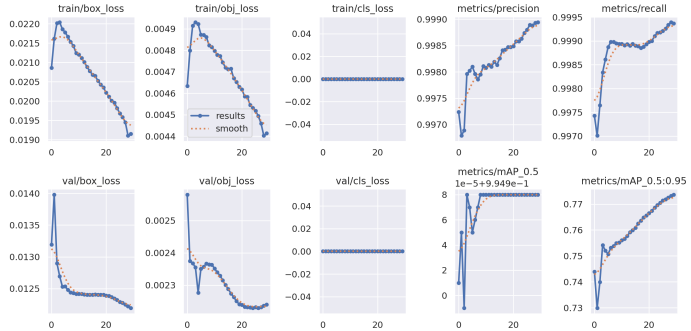
4 Detection: YOLOv5

Example of a validation batch with **labels**

Example of a training batch with **predictions**

Training and Validation Results

4 Detection: YOLOv5



- **Precision** and **recall** above 99.8%;
- **mAP@0.5** close to 1.0, while **mAP@0.5:0.95** shows consistent growth;
- **Class loss is zero**: expected, as there is no multi-class classification.

Metrics at Epoch 30

4 Detection: YOLOv5

- **Train Loss:**
  - Box loss: 0.01915
  - Obj loss: 0.00441
- **Validation Loss:**
  - Box loss: 0.01219
  - Obj loss: 0.00224
- **Metrics:**
  - Precision: 0.99894
  - Recall: 0.99937
  - mAP@0.5: 0.99498
  - mAP@0.5:0.95: 0.77362

Splitted test folder results

4 Detection: YOLOv5

Test folder	Elements	Precision	Recall	mAP50	mAP50-95	Mean IoU	IoU>0.7 %	FPS
db	10132	0.9610	0.9340	0.9670	0.5460	0.7615	71.64	75.72
blur	20611	0.9850	0.9750	0.9880	0.6490	0.8178	87.10	73.06
fn	20967	0.9680	0.9690	0.9800	0.5880	0.8013	85.41	77.16
rotate	10053	0.9930	0.9950	0.9940	0.6840	0.8303	96.23	79.82
tilt	30216	0.9910	0.9900	0.9940	0.5800	0.7831	86.17	75.06
weather	9999	0.9990	0.9990	0.9950	0.7760	0.8771	98.46	76.53
challenge	50003	0.9850	0.9870	0.9920	0.6720	0.8343	93.52	63.75

Test results and comparison with other methods

4 Detection: YOLOv5

Method	Overall Accuracy	Base (100k)	DB (20k)	FN (20k)	Rotate (10k)	Tilt (10k)	Weather (10k)	Challenge (10k)	Blur (20k)	Speed (FPS)
Faster RCNN	92.9	98.1	92.1	83.7	91.8	89.4	81.8	83.9	-	17.6
YOLO9000	93.1	98.8	89.6	77.3	93.3	91.8	84.2	88.6	-	43.9
SSD300	94.4	99.1	89.2	84.7	95.6	94.9	83.4	93.1	-	40.7
TE2E	94.2	98.5	91.7	83.8	95.1	94.5	83.6	93.1	-	3.2
RPnet	94.5	**99.3**	89.5	85.3	94.7	93.2	84.1	92.8	-	85.5
YOLOv5	**96.7**	97.2	**97.7**	**92.9**	**98.9**	**98.9**	**99.0**	90.6	-	**218.3**
Our YOLOv5	86.99	77.36	71.64	85.41	96.23	86.17	98.46	**93.52**	**87.10**	74.44

Comparison of different networks on CCPD test sets. In bold the best result and underlined the second one.

Test results - 1

4 Detection: YOLOv5

Blur sample

Challenge sample

Dark/bright sample

Far/Near sample

Test results - 2

4 Detection: YOLOv5

Rotate sample

Tilt sample

Weather sample

Table of Contents

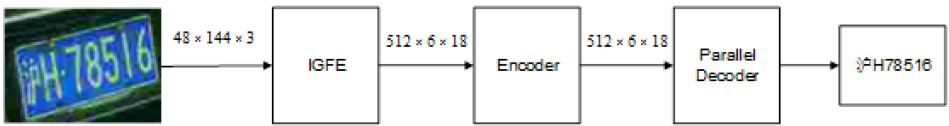
5 Recognition: PDLPR

Character Recognition Architecture

5 Recognition: PDLPR

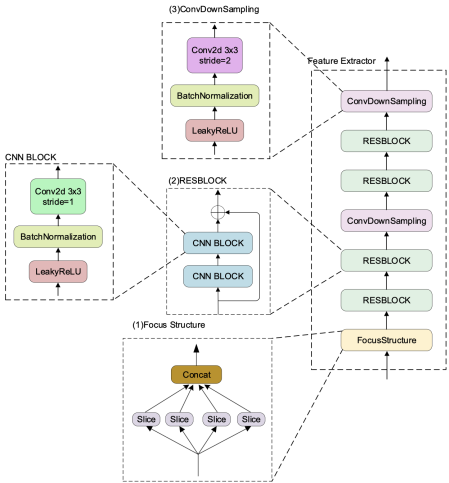
The model proposed in the paper, called **PDLPR**, consists of three main modules:

- **IGFE (Improved Global Feature Extractor)**: extracts global features from the license plate image (initial size 48×144×3), converting it into a feature map of size 512×6×18.
- **Encoder**: incorporates position information (Positional Encoding) and applies **Multi-Head Attention**.
- **Parallel Decoder**: decodes the sequence.



IGFE Module Implementation

5 Recognition: PDLPR



Specifications

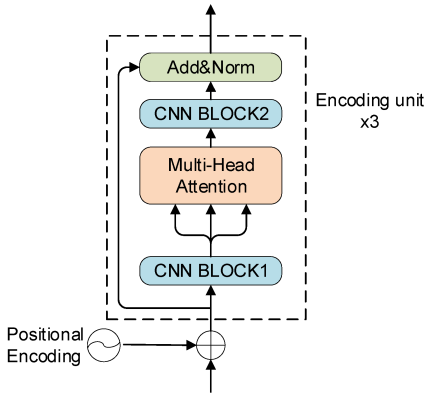
- **Layer order in convolutional blocks:**
  - Conv2D → Batch Norm → Leaky ReLU
- **Padding in Conv2D:**
  - Our model: padding=1
  - Paper: padding=0
- **Channels in ConvDownSampling Blocks:**
  - from 12 to 256 (first downsampling)
  - from 256 to 512 (second downsampling)

Encoder Implementation

5 Recognition: PDLPR

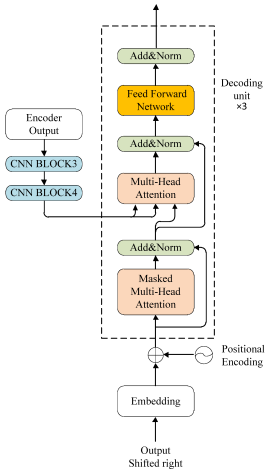
Specifications

- **Initial Flattening:** The 3D sequence (B, 512, 6, 18) output from the IGFE module is flattened to obtain a sequence (B, 108, 512);
- **1D Convolution (CNNBlock 1 & 2)**
- **Final Image Reconstruction:** After the Encoder, the sequence is transformed back to its original shape (B, 512, 6, 18).



Decoder Implementation

5 Recognition: PDLPR



Specifications

- **2D Convolutions (CNN Block 3 & 4)**;
- **Flattening:** after CNN Block 4, the features (B, 512, 6, 18) are flattened into a sequence (B, 108, 512);
- **Input to Self-Attention:** target_seq is used as **Query**, **Key**, and **Value**;
- **Input to Cross-Attention:**
  - **Query:** output of Add&Norm 1;
  - **Key and Value:** the flattened sequence from the spatial features processed by the Encoder and CNN BLOCK.

target_seq in the Decoder

- **Initialization:** target_seq is a sequence of dummy tokens (random values), each with shape (1, 7, 512).
- **Self-Attention:** target_seq interacts with itself through the Self-Attention module.
- **Cross-Attention:** target_seq is combined with information from the Encoder. Thanks to Self-Attention and Cross-Attention, the sequence evolves gradually and is refined to correctly represent the license plate.

Final License Plate Prediction

5 Recognition: PDLPR

- After the final decoder unit, the target_seq contains the complete representation of the license plate, enriched with spatial information from the Encoder.
- The final target_seq is passed through a **projection head**, which generates logits for each character of the license plate.
- The logits are interpreted as probabilities for each character. The maximum values for each of the 7 positions on the license plate, corresponding to province, alphabet, and numbers, are extracted.
- The **decoding** of the license plate is done by separating the province, alphabet, and numbers, based on the predicted character positions.
- The final result is the **reconstructed license plate** in a readable format.

Logit and Projection Head

- The **projection head** is a Linear(512, total_chars) layer applied to each vector in the target_seq.
- For each position of the license plate, the head generates a vector of size equal to the total number of characters (total_chars).
- The result is called a logit: raw numerical values representing the activation for each possible character.
- Given a logit, the following is applied:
  - `argmax`: to select the most probable character.
- Finally, the license plate is reconstructed by assembling the 7 predicted characters.

Image Preprocessing for PDLPR

5 Recognition: PDLPR

- **Cropping and resizing the license plate**:
  - The license plate is cropped based on a provided bounding box;
  - The image is then resized to a fixed size of (48x144) using bilinear interpolation.
- **Augmentation**:
  - **Color jitter**: Random variations in brightness (0.2), contrast (0.2), saturation (0.2), and hue (0.1);
  - **Gaussian noise**: Random noise is added with $\mu = 0$ and $\sigma = 0.01$.

Training

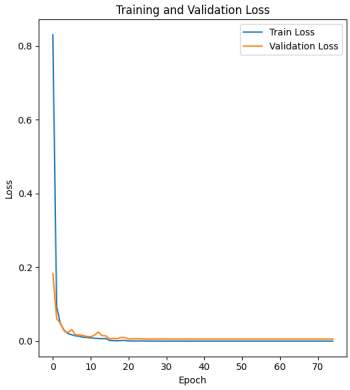
5 Recognition: PDLPR

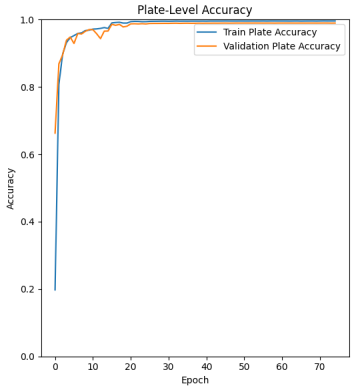
- **Batch size**: 128
- **Epochs**: 75 (vs 1000 in the paper)
- **Loss Function**: Connectionist Temporal Classification
- **Optimizer**: Adam
- **Initial Learning Rate**: $1 \cdot 10^{-3}$
- **Training Time**: 11 min per epoch (train + validation)

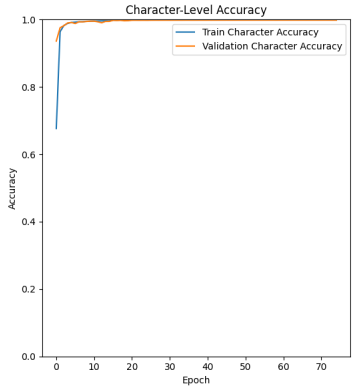
Examples of training images

Training and Validation Results

5 Recognition: PDLPR

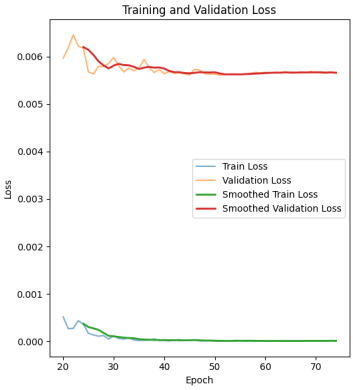


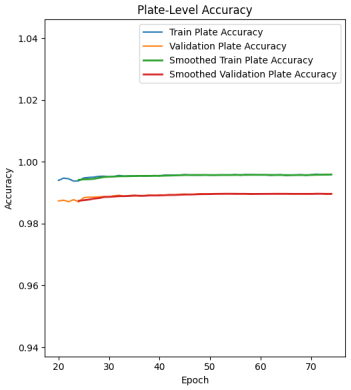


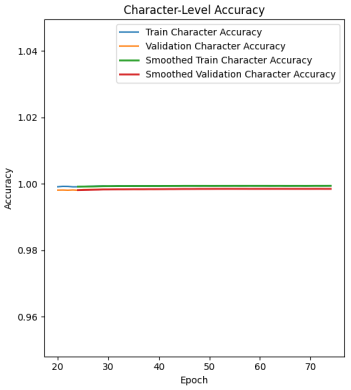


Training and Validation Results - 2

5 Recognition: PDLPR







Metrics at Epoch 75 (best model)

5 Recognition: PDLPR

- **Train Loss:** 9.61e-06
- **Validation Loss:** 0.0056
- **Metrics:**
  - **Plate Accuracy**: 98.96
  - **Character Accuracy**: 99.84

Test Results and Comparison with Other Methods (CCPD)

5 Recognition: PDLPR

Method	Overall Accuracy	Base (100k)	DB (20k)	FN (20k)	Rotate (10k)	Tilt (10k)	Weather (10k)	Challenge (10k)	Blur	Speed (FPS)
Zou et al.	97.8	99.3	98.5	98.6	92.5	96.4	99.3	86.6	-	-
Zhang et al.	98.5	99.6	98.8	98.8	96.4	97.6	98.5	88.9	-	40.2
Zhang et al.	98.9	99.8	99.2	99.1	98.1	98.8	98.6	89.7	-	40.2
Qin et al.	97.2	99.3	92.9	93.2	97.9	95.5	98.8	92.4	-	36.0
Qin et al.	97.6	99.5	93.3	93.7	98.2	95.9	98.9	92.9	-	26.0
Fan et al.	98.8	99.7	99.1	99.0	99.1	99.0	98.5	88.0	-	11.7
Fan et al.	99.0	99.8	99.2	99.2	**99.6**	**99.6**	98.5	88.8	-	26.0
PDLPR	**99.4**	**99.9**	**99.5**	**99.5**	99.5	**99.4**	**99.4**	**94.1**	-	**159.8**
Our PDLPR	66.05	98.96	56.38	60.63	54.92	41.74	97.32	64.16	54.32	108.88

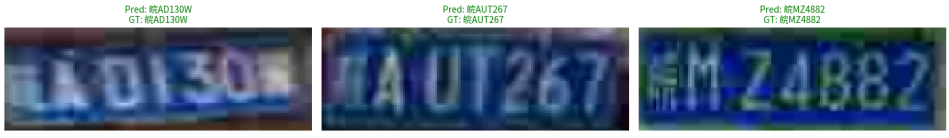
Comparison of performance on CCPD test subsets. Best results in bold, second best in underlined.

Prediction Examples on CCPD Challenge and Weather

5 Recognition: PDLPR

Predizioni - ccpd_challenge.csv

Predizioni - ccpd_weather.csv



Examples of correct and incorrect predictions on subsets of the CCPD dataset.

Prediction Examples on CCPD Tilt and Fn

5 Recognition: PDLPR

Predizioni - ccpd_tilt.csv

Predizioni - ccpd_weather.csv

Examples of correct and incorrect predictions on subsets of the CCPD dataset.

Table of Contents

6 Conclusion

Faster-RCNN + STAR-Net vs Yolov5 + PDLPR: Pipeline Comparison

6 Conclusion

Feature	Faster-RCNN + STAR-Net	Yolov5 + PDLPR
Goal	Accurate	Fast
Recognition: Geometric pre-processing	Required	None
Feature Extraction / Modeling / Decoding	Heavy	Light
Inference Speed	Slow	Fast
Robustness	High	High
Scalability	Limited	Scalable
CCPD Performance	Accurate but Slow	Accurate and Fast

Table of Contents

Bibliography

7 Bibliography

Baek, Y., Kim, B., Lee, D., Park, S., Han, G., Yun, S., & Lee, H. (2019). What is wrong with scene text recognition model comparisons? dataset and model analysis. *arXiv preprint arXiv:1904.01906*.

Tao, L., Hong, S., Lin, Y., Chen, Y., He, P., & Tie, Z. (2024). A Real-Time License Plate Detection and Recognition Model in Unconstrained Scenarios. *Sensors*, 24(9), 2791.

Xu, Z., Yang, W., Meng, A., Lu, N., Huang, H., Ying, C., & Huang, L. (2018). Towards end-to-end license plate detection and recognition: A large dataset and baseline. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany, 8–14 September 2018.

Prajapati, R. K., Bhardwaj, Y., Jain, R. K., & Hiran, D. K. K. (2023). A Review Paper on Automatic Number Plate Recognition using Machine Learning: An In-Depth Analysis of Machine Learning Techniques in Automatic Number Plate Recognition: Opportunities and Limitations. In *2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN)*, Ghaziabad, India, 527–532.

License Plate Detection and Recognition

Thank you for listening!

Any questions?