Comparative Analysis of Traditional Methods and Deep Learning Approaches in License Plate Recognition

George Wang, Mutian Ling, Xichu Xiao, Chung-Chun Wang

December, 2023

Abstract

License Plate Recognition (LPR) plays a pivotal role in applications ranging from traffic monitoring to law enforcement and automated toll collection. This study conducts a comparative analysis of LPR, contrasting traditional image processing techniques with advanced deep learning approaches. Starting with conventional Tesseract optical character recognition (OCR) methods, the research covers image pre-processing, segmentation, character recognition, and template matching, emphasizing their interpretability and control. The exploration then extends to modern OCR methods, including Tesseract OCR's Long Short-Term Memory (LSTM) based forms, highlighting LSTM's improved handling of complex text layouts. Subsequently, a Convolutional Neural Networks (CNN) based model and an End-to-End CNN model for LPR are developed. Intriguingly, these models, while proficient in feature extraction and pattern recognition, does not surpass traditional methods in speed and accuracy in our tests. This outcome underlines the complexities of applying deep learning to specific real-world scenarios. The comparative study evaluates these approaches, revealing that while deep learning approaches offer robustness and adaptability, traditional methods maintain relevance through their precision and interpretability in certain contexts. This research aims to inform the development of LPR systems, balancing traditional and deep learning techniques to meet the diverse needs of intelligent transportation systems.

Contents

1 Introduction

License Plate Recognition (LPR) technology has become increasingly vital in addressing various social issues and safety concerns. Its relevance extends from traffic management to addressing critical issues such as the Amber Alert system in the United States, hit and run incidents, and the widespread deployment of surveillance cameras for public safety.

The Amber Alert system, initiated to rapidly disseminate information about missing and abducted children, heavily relies on the efficient identification of vehicles. LPR plays a crucial role in this system by enabling the swift and accurate recognition of license plates, which can be pivotal in locating missing children and preventing potential tragedies.

Similarly, the unsettling prevalence of hit and run incidents, where drivers flee the scene of an accident, poses a significant challenge for law enforcement. LPR systems are instrumental in these scenarios, offering a technological means to trace offending vehicles swiftly, thereby aiding in bringing perpetrators to justice and providing closure to victims and their families.

Moreover, the proliferation of surveillance cameras in urban environments has opened new avenues for utilizing LPR. These cameras, constantly monitoring public spaces, generate vast amounts of visual data. Efficient LPR systems can analyze this data to enhance public safety, manage traffic flows, and even aid in urban planning.

However, the effectiveness of LPR systems hinges on their accuracy and adaptability to different conditions, such as varying light, weather, and camera angles. This study aims to compare traditional image processing techniques with advanced deep learning approaches, specifically Convolutional Neural Networks (CNN), in the context of LPR.

This report presents an in-depth exploration of four distinct LPR methodologies, each with its computational techniques and strategies. We begin with the Traditional LPR method, grounded in time-tested image processing principles, employing location processing, contour detection, character segmentation and other multi-step process that culminates in character recognition with Tesseract OCR. Moving forward, we introduce a hybrid method that combines Recurrent Neural Networks (RNN) with traditional computer vision techniques, capitalizing on the robustness of deep learning for character classification post the initial processing stages.

Incorporating the strengths of deep learning, our study also examines a CNN-based model, adapted from an open-source project for Indian license plates and fine-tuned for the nuances of Turkish license plates.

Our investigation then leads to a more sophisticated LPR system that harnesses the sequential processing power of Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, combined with a CNN model, and employs Connectionist Temporal Classification (CTC) loss for its effectiveness in sequence recognition. This advanced method is specifically tailored to address the variable-length and sequential nature of text in license plates, integrating comprehensive stages from data preparation to feature extraction and sequence learning.[7]

The versatility of LPR systems is tested against the challenges of varying environmental conditions, aiming to provide a balanced perspective that underscores the evolution of LPR technologies. By comparing these methodologies, our goal is to critically assess their performance, efficiency, and robustness, offering insights into their potential for enhancing community safety and aiding law enforcement. Through this analysis, we seek not only to advance LPR as a technological pursuit but also to underscore its value as a tool for social welfare, paving the way for more effective and adaptable systems in the future.

2 System Design/Methodology

2.1 Dataset Description

Before delving into the specifics of the methodologies employed in our study, it is essential to describe the dataset that forms the foundation of our License Plate Recognition (LPR) system evaluation.

We utilized the Synthetic Turkish License Plates dataset, available on Kaggle, which comprises 10,000 images of various vehicle license plates. This dataset was specifically chosen for its diversity, representing a wide array of environmental conditions that a robust LPR system is likely to encounter. The images within this dataset capture a range of scenarios, including different lighting conditions, angles, and weather situations, providing a comprehensive basis for assessing the adaptability and accuracy of LPR techniques.

Each image in the dataset is annotated with the corresponding license plate text, allowing for precise ground truth comparison. These annotations play a critical role in evaluating the performance of the LPR methodologies in terms of recognition accuracy.[8]

With the dataset defined, we will now proceed to detail the LPR methods evaluated in this study.

Figure 1: Example Figure from Dataset

2.2 Traditional LPR Approach

Our study's first method employs a Traditional License Plate Recognition (LPR) approach, which combining elements of computer vision and Optical Character Recognition (OCR). This method involves a sequence of steps, each designed to progressively refine the image and extract the necessary information for character recognition.

Loading and Preprocessing: The process commences with the loading of an image using OpenCV's $cv2.imread$ function. After ensuring the image is correctly loaded, it is converted to grayscale, which simplifies the subsequent processing stages by reducing the complexity of the image data.

Edge Detection using Sobel Operator: Edge detection is pivotal in identifying the distinct edges of a license plate. We apply the Sobel operator in both horizontal and vertical directions, which accentuates the edges in the image, crucial for recognizing the rectangular shape of a license plate.

Thresholding and Noise Reduction: We employ Gaussian blur for mitigating noise, followed by Otsu's thresholding method (using cv2.threshold). This step converts the image into a binary format, streamlining the feature and contour extraction processes.

Morphological Operations: To refine the image, we perform morphological operations, specifically dilation and erosion. These operations are instrumental in connecting disjointed parts (such as characters on a license plate) and eliminating minor, irrelevant details.

Contour Detection and License Plate Extraction: Utilizing cv2.findContours, the script identifies various contours in the image. It focuses on locating a contour that aligns with the aspect ratio of a typical license plate, which is then isolated for further processing.

License Plate Recognition using Tesseract OCR: The extracted license plate region is processed using Tesseract OCR ($pytesseract.image\_to\_string$), a leading open-source OCR engine. Tesseract OCR was initially developed by HP, and later improved by Google. It's renowned for its high accuracy in converting images of text into machine-readable text formats. To enhance recognition accuracy, a

whitelist of characters comprising numbers and uppercase letters is specified. It should be noted that the configuration of Tesseract OCR we use here is '–oem 0', which means that we only use its original model matching engine.

Figure 2: Tesseract OCR Icon

Optional Post-processing: The code includes provisions for additional post-processing steps, such as visualizing intermediate results using matplotlib.pyplot and refining OCR outcomes with regular expressions, although these are not mandatory.

The methodology is operationalized using two primary Python scripts: *traditional_LPR.py* and *operation.py*. The former encapsulates the core steps of LPR - from image loading to character recognition - while the latter script focuses on batch processing of images, accuracy assessment, and error analysis. The scripts are designed to process a predefined number of images from a specified directory, identify the text on the license plates, and compare the OCR results with the correct answers extracted from the file names.

The effectiveness of this methodology is evaluated based on multiple criteria, including the accuracy of character recognition across a diverse range of images and environmental conditions, as well as the system's processing speed and robustness under varying scenarios.

2.3 Deep Learning Approach via RNN

The methodology for the second LPR approach, focusing on Long Short-Term Memory (LSTM), includes initial steps similar to the Traditional LPR method but differs in the OCR phase. Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) specialized in processing sequences of data, making them ideal for tasks involving time series, natural language processing, and character recognition in OCR. After dataset preprocessing, the Tesseract OCR is configured to utilize its LSTM-based engine (–oem 1) for character recognition. This setting aims to improve the accuracy of character recognition by leveraging LSTM's capability to understand context and sequence in text, particularly useful in complex scenarios. It should be noted that if we do not specify the OEM mode, Tesseract will use the default engine (LSTM) to perform OCR operations.

2.4 Deep Learning Approach via CNN

The third method of License Plate Recognition (LPR) synergizes Convolutional Neural Networks (CNNs) with classic computer vision techniques for preprocessing. For the CNN-based LPR model, we adapted an existing open-source neural network architecture originally designed for Indian license plate recognition.[2] This model, sourced from a GitHub repository, was modified and fine-tuned to suit the specific features and patterns of the license plates in our dataset. The adaptation involved adjusting the network layers and parameters to optimize performance for the Turkish license plates in the Kaggle dataset. This methodology is segmented into several distinct stages, as outlined below:

1. Preprocessing and Character Segmentation: Extraction of License Plate: The process initiates with the use of OpenCV to preprocess the image and extract the license plate. This extraction is executed through a predefined function extract_license_plate, which is similar to the one employed in the Traditional LPR method. This function effectively isolates the license plate from the rest of the

image. Segmentation of Alphanumeric Characters: The segmented characters from the license plate are identified using contour detection. The find_contours function converts the image to grayscale, applies thresholding, and then employs cv2.findContours to detect contours, which are then resized, formatted, and stored for further processing.

2. CNN Model for Character Recognition: Model Architecture: A CNN model is adaped from a model for Indian plate updated by Sarthak Vajpayee, specifically designed for classifying segmented characters. The architecture comprises convolutional layers (Conv2D), max pooling layers (MaxPooling2D), dropout layers (Dropout) for regularization, and dense layers (Dense) for final classification. Class Categories: The network is calibrated to classify 36 classes, encompassing digits and uppercase letters, which aligns with the standard characters found on most license plates.[6]

3. Data Augmentation and Training: Implementation of Data Augmentation: To enhance the model's robustness and adaptability to variations in the input data, data augmentation is employed using ImageDataGenerator. This technique introduces alterations like shifts and rescales in the training data. There are a total of 36 categories in these training data. Each category corresponds to a letter or number and has pictures of characters in different angles and other environments. Setting up Generators: The script establishes training and validation generators, which direct the flow of data from specific directories containing training and validation images.

4. Model Training and Callbacks: Training Process: The model is compiled and trained using the fit_generator method. This process is augmented with callbacks for early stopping (stop_training_callback) and logging (tensorboard_callback), enhancing the training efficiency. Training Continuation Criteria: Training is continued until either the validation accuracy surpasses a predetermined threshold or the maximum number of epochs is reached.[5]

5. Character Recognition and Aggregation: Predicting and Assembling Characters: The show_results function processes the segmented character images to fit the input shape of the CNN. Subsequently, the trained model predicts the characters, which are then aggregated to form the complete license plate number. The implementation of this methodology is encapsulated within two Python scripts: *train.py* and *CNN_test.py*. The *train.py* script is responsible for character segmentation, model creation, training, and prediction. Meanwhile, *CNN_test.py* focuses on processing the images from a specified folder, applying the model for prediction, and evaluating the performance based on accuracy and processing time.

This approach effectively combines the precision of traditional image processing methods for initial segmentation with the advanced capabilities of deep learning for character recognition. It is a robust method to tackle the complex task of license plate recognition, leveraging the strengths of both domains to achieve high accuracy in identifying license plate characters.

2.5 Deep Learning Approach via End-to-End CNN

The last approach in our study implements an end-to-end License Plate Recognition (LPR) system using a combination of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) networks. This methodology is adept at handling the sequential nature of text in license plates and consists of the following stages:[1]

1. Data Preparation: Custom Data Generation: The approach utilizes a CustomDataGenerator to manage image data efficiently. This generator is essential for handling large datasets and feeding them into the model in batches. Data Retrieval: Training and validation data paths are retrieved using the get_data function, ensuring structured access to the image data.

2. Model Architecture: Input and CNN Layers: The model starts with an input layer to accommodate the license plate images. Subsequent CNN layers (Conv2D and MaxPooling2D) are applied for feature extraction. These layers are instrumental in detecting various patterns, edges, and other features in the images. RNN Layer and Reshaping: Post-CNN, the model incorporates a reshaping layer to adapt the CNN output for the RNN layers. An RNN layer, specifically an LSTM, is then

used to understand and process the sequence of characters in the license plates. Output Layer and CTC Loss Function: The final layer is a dense layer with softmax activation, designed to predict a fixed number of classes (characters on the license plate). The model employs a CTC (Connectionist Temporal Classification) loss function, which is particularly effective for sequence recognition tasks with variable timing.[4]

3. Model Training: Compilation and Training: The model is compiled and trained using model.fit, with defined parameters for batch size, epochs, steps per epoch, and validation steps. Custom callbacks for early stopping or checkpoints are incorporated to optimize the training process.

4. Prediction and Decoding: Model Predictions: After training, the model predicts labels on the validation set. Decoding Function: A custom function decode_batch_predictions is included to convert raw predictions into human-readable text using CTC decoding. This function maps the predictions to actual characters, forming the license plate numbers.

5. Performance Evaluation: Accuracy Calculation: The script calculates accuracy by comparing predicted labels with true labels extracted from the validation image file names. Optionally, the Character Error Rate (CER) can be calculated using Levenshtein distance, providing a measure of similarity between the predicted and true text.

6. Visualization and Analysis: Intermediate Layer Outputs (Commented Out): The code includes sections for visualizing intermediate layer outputs and activation, offering insights into the model's behavior and aiding in debugging.

This methodology capitalizes on the strengths of CNNs for spatial feature extraction and RNNs for sequential data processing. The use of CTC loss is particularly suitable for LPR tasks, where the alignment between the input image and the text label is not fixed. The implementation is carried out through scripts *imagepath.py*, *datagenerator.py*, and $CNN\_LPR.py$, encompassing data preparation, model architecture, training, and evaluation stages. This sophisticated approach aims to achieve high accuracy in recognizing license plate characters, even under varying environmental conditions.

3 Evaluation

3.1 Traditional LPR Approach

In evaluating the Traditional LPR method using Tesseract OCR, the accuracy rate stands at 89.10% for a dataset of 1000 images, with a total runtime of 158.95 seconds. While the system correctly identified 891 license plates, it showed limitations in distinguishing between similar characters, such as '0' and 'O', or '1' and 'I'. These errors indicate areas for improvement, particularly in character segmentation and recognition accuracy under varying conditions. The evaluation highlights the method's overall efficiency and accuracy, but also underscores the need for enhancements to address specific character recognition challenges.

```
Correct: 78FS553, Identified: 78F8553
Correct: 78OS039, Identified: 78OSO39
Correct: 79GZ690, Identified: 7902690
Total images processed: 1000
Correctly identified: 891
Accuracy: 89.10%
Total runtime: 158.95 seconds
```

Figure 3: Screenshot of Traditional LPR log file

3.2 Deep Learning Approach via RNN

The evaluation of the LSTM-based LPR approach shows a notable improvement in accuracy, achieving 95.30% on a dataset of 1000 images with a total runtime of 136.92 seconds. This method, which correctly identified 953 license plates, demonstrates the effectiveness of LSTM in OCR, particularly in recognizing complex character sequences. However, there were still instances of misinterpretation, mostly involving similar characters. The increased accuracy compared to the traditional method indicates the potential of LSTM in enhancing character recognition accuracy in LPR systems.

```
Correct: 78L1152, Identified: 7841152
Correct: 78OI187, Identified: 78O1187
Correct: 79I0877, Identified: 7910877
Correct: 79I4182, Identified: 7914182
Total images processed: 1000
Correctly identified: 953
Accuracy: 95.30%
Total runtime: 136.92 seconds
```

Figure 4: Screenshot of LPR with LSTM log file

3.3 Deep Learning Approach via CNN

The CNN model adapted for Turkish license plates showed an accuracy of 47.70% and an average processing time of 0.33 seconds per image over 1000 images. This performance is notably lower in

accuracy compared to both the traditional and LSTM methods, and it's also slower than the LSTM approach. This suggests that while the model processes images relatively quickly, it struggles significantly with accurate recognition, highlighting the need for further model optimization and training specific to the dataset characteristics.

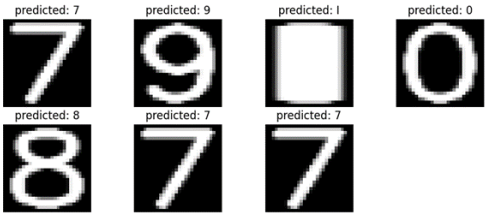
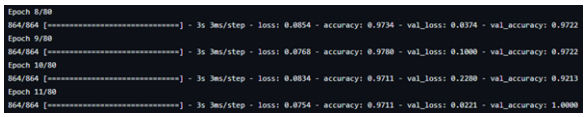
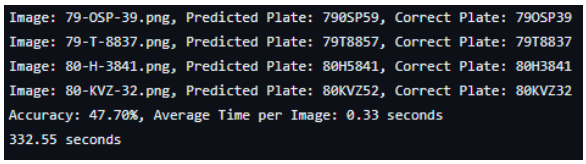


Figure 5: Predictions for Segmented Characters



(a) CNN LPR Training log file



(b) CNN LPR Testing log file

Figure 6: Screenshots of Regular CNN LPR log file

3.4 Deep Learning Approach via End-to-End CNN

The end-to-end CNN model's performance in this evaluation indicates significant challenges. With an accuracy of 0.00% over 1000 processed images, the model failed to correctly identify any license plates. The extremely fast total run-time of 0.11 seconds suggests that while the model is quick, it lacks effectiveness in accurate recognition. The neuron activation patterns indicate issues such as characteristic overfitting. This outcome might point to issues such as inadequate training, model complexity not aligning with the task, or improper preprocessing steps. These results highlight a critical need for revisiting the model's architecture and training process.

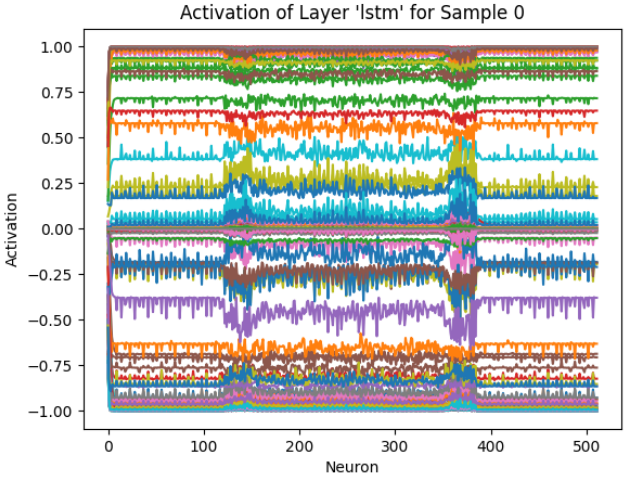


Figure 7: The activation of a neuron in the LSTM layer over time

```
Accuracy: 0.00%
Total runtime: 0.11 seconds
Total number of processed images: 1000
```

Figure 8: Screenshot of End-to-End CNN log file

4 Conclusions and Recommendation

4.1 Conclusions

Our research shows that, on the premise of using the Tesseract open source engine, both the traditional model matching license plate recognition method and the method using RNN can have better accuracy and speed. Because of the high robustness and adaptability of neural networks, the method using LSTM has the highest accuracy and speed.

The ordinary CNN model method is inferior to LSTM and traditional methods in terms of performance indicators because it is adapted from another open source project. We speculate that the reasons for the poor accuracy and speed of the model may be: 1. The number of neural network layers in the model itself is too basic, making it unable to handle more complex and varied license plate data. 2. The difference between the training database and the testing database is too large, resulting in unsatisfactory test results.

The above three methods all rely on image preprocessing, especially the accuracy of character segmentation. If the character segmentation accuracy is not enough, the performance indicators of these three methods will decrease to varying degrees.

For the end-to-end CNN model method, although it has the lowest accuracy among the four methods, this method shows extremely fast processing speed, indicating that it still has a lot of room for development. We speculate that it may due to the model's neural network layers are too few, causing the model to be unable to discover and remember sufficient feature characteristics from the training database, and overfitting occurs; another possible reason is the training dataset does not have enough samples. In order to increase the complexity and accuracy of the model, the dataset requires about 40 billion images. Due to capacity constraint of the hardware, the database we trained is only 10,000 images.

Therefore, the project concludes that limitations in dataset size and network depth likely hindered

the performance of the neural network-based methods. Additionally, the traditional approach benefits from mature image processing techniques, incorporating machine learning strategies that enhance both efficiency and accuracy, also is more interpretive. At the same time, it should be pointed out that the performance of the license plate recognition method is also related to the hardware environment. The better the hardware conditions, the faster the running speed.

4.2 Recommendation

Based on the findings of our comparative study on License Plate Recognition (LPR) techniques, we recommend the following:

- Further Research on Deep Learning Models: Explore more advanced deep learning architectures and training techniques to enhance the accuracy and speed of CNN-based LPR systems.
- Dataset Expansion: Collect and integrate a more diverse set of images, including different lighting and weather conditions, to improve the robustness of LPR models.
- Application Development: Investigate the potential of integrating these LPR technologies into broader applications like traffic management systems and public safety networks.[3]

5 Acknowledgements

We would like to extend our heartfelt gratitude to several individuals and organizations:

- Professor J. Matias Di Martino, Professor Guillermo Sapiro and all TAs for ECE 588 at Duke University for their invaluable guidance and support throughout this project.
- Our team members for their contributions and collaborative efforts.
- The developers of the open-source CNN model and Tesseract OCR, which were integral to our comparative study in LPR technologies.
- Kaggle for providing the dataset that was instrumental in our study.
- All those who provided technical support and resources, making this research possible.

This study is not only a reflection of our efforts but also a testament to the collaborative spirit of the academic and open-source communities.

References

[1] github. license-plate-recognition, 2022.

[2] SarthakV7. Ai-based-indian-license-plate-detection, 2021.

[3] J. Shashirangana, H. Padmasiri, D. Meedeniya, and C. Perera. Automated license plate recognition: A survey on methods and techniques. *IEEE Access*, 9:11203–11225, 2021.

[4] A. SINGH. License plate recognition final, 2020.

[5] S. Vajpayee. Ai-powered indian license plate detector., 2019.

[6] S. VAJPAYEE. License plate recognition using cnn, 2020.

[7] M. D. C. Velarde and G. Velarde. Benchmarking algorithms for automatic license plate recognition, 2021.

[8] T. ÜSTÜNKÖK. Synthetic turkish license plates, 2019.