Robust Efficient License Plate and Character Detection System Based on Simplified CNN

Selena He
Kennesaw State University
Marietta, Georgia, USA
she4@kennesaw.edu

Tu N. Nguyen
Kennesaw State University
Marietta, Georgia, USA
tu.nguyen@kennesaw.edu

Kun Suo
Kennesaw State University
Marietta, Georgia, USA
ksuo@kennesaw.edu

ABSTRACT

Current license plate recognition systems struggle with image noise reduction and license plate feature detecting processes. This paper presents an efficient and highly accurate license plate detection and character detection program based on the YOLO neural network, which is a simplified CNN-based neural network frame for robust image processing systems. Different than most approaches, the system we proposed simply requires a prioritized analysis of the dataset in order to evaluate potential noises inside images so that program implementations could be more effective and more targeted to design and optimize with YOLO neural network. With our presented system, the accuracy of license plate detection improves from 63% which is performed by traditional image processing methods to 90.3%.

CCS CONCEPTS

• **Computing methodologies → Machine learning**; *Machine learning Approaches*; *Neurual Networks*; • **Neural Networks →** CNNs.

KEYWORDS

Convolutional Neural Networks (CNN), License Plate Recognition, YOLO, Big Data Analytics, Deep Learning Model

ACM Reference Format:
Selena He, Tu N. Nguyen, and Kun Suo. 2023. Robust Efficient License Plate and Character Detection System Based on Simplified CNN. In *2023 ACM Southeast Conference (ACMSE 2023), April 12–14, 2023, Virtual Event, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3564746.3587108

1 INTRODUCTION

Vehicle License Plate Recognition is highly demanded nowadays with the growth of people's daily motor transportation. With future considerations, the license plate recognition function is also critical for monitoring self-driving systems, implementing smart cities, etc. As Anagnostopoulos et al. [5, 8] researched, however, there are limited methods for commercial use of license plate recognition, and optimizations of such image processing programs are hard to implement. One big challenge of accomplishing the system is to reduce noises in images, as the noises that appear in the image could be various and cause issues for feature detection. Usually, license plate recognition systems are set up with a general image feature frame detection algorithm [5], and then researchers will optimize the programs depending on datasets. While such an approach is truly feasible, it requires a lot of experimentation and optimization, and there is a certain degree of randomness. Once the optimization effect is not ideal, testers may even overturn the entire program. Considering the remaining issue, we attempted to analyze our license plate dataset before implementing the system and thus made highly effective progress in our experiments.

Other than analyzing datasets, our license plate detection program was improved with a simplified Convolutional Neural Network (CNN). The common ways to extract license plate features from images are based on image processing algorithms to reduce essential noises and emphasize license plate frames [4, 13]. As we mentioned before, however, noises that exist in all images through a dataset remain erratic, and without deep learning models, the program can hardly adjust its approach to processing input images properly. Therefore, we implement a single neural network on training the license plate detection dataset based on the YOLO framework [16].

Also, when approaching the character recognition process, the common way is to calculate black and white pixels from a binary image, thus getting potential outputs as single characters for the next step recognition [10]. Although efficient, such an approach could not easily achieve perfect accuracy in a large number of datasets. If the cropped outputs, which are generated by pixels calculating only, went wrong, it would be nearly impossible for the character recognition program to do the work correctly. Therefore, inspired by the same method we used in license plate detection, we implement a training model as we did the same in the character recognition system, and use the model twice to traverse through images, look for potential features, and thus crop and evaluate each output.

In summary, the contributions of this paper present as follows:

- An approach of optimizing noise reduction by analyzing the dataset's general features;
- An efficient convolutional neural network used for license plate detection, which improves the common feature detection accuracy from 63% to 90%;
- An improved character segmentation process which is based on the same training model as character recognition, using such a model could greatly improve the accuracy out of cropping characters through binary pixels.

This paper will be organized as follow: in Section 2 we summarize the existing License Plate Detection, Character Segmentation, and

Recognition processes. In Section 3 we present the contributions in detail with our license plate and character detection system. In Section 4 we evaluate the proposed system with experiments and present comprehensive results and comparative results of the existing methods. Section 5 concludes the paper.

2 RELATED WORK

As deep learning is in general use in recent years, there are a lot of new methods to accomplish the license plate recognition system. Before deep learning, one of the most common ways to detect license plates is using image processing [5]. The implementation is to use general image processing to reduce noise and emphasize the edges of license plates in the image. In Figure 1, a data image is converted into a binary scale, and the program used a Gaussian filter to reduce noise and emphasize edges with opening and closing operations. License plates could be extracted with a highlight box,

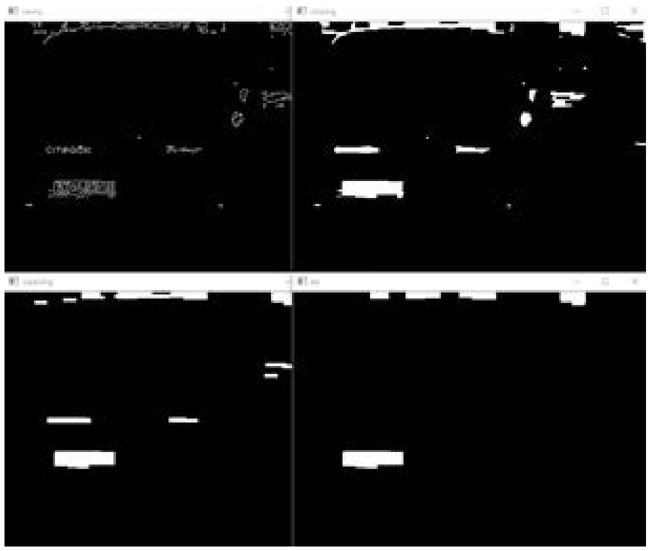

Figure 1: Extract Plate Features with Gaussian Noise-reducing Filter for Edge Emphasizing

and can be detected with certain aspect ratios. Some programs will also implement a strong aspect ratio model which conducts possible features of a license plate. For example, as Bulan et al. [3] proposed, if the license plate in an image shows as tilted, the program could detect certain possibilities and thus process the image and try to covert the license plate back.

Other than image processing, Bulan et al. [3] implemented an automatic detection system with failure identification based on the CNN classifier. The program will locate potentially identified regions along with their failure rankings. One problem that remains is that the program would require two CNN-based classifiers to detect potential regions and analyze failure scales. Even though the accuracy improves, the efficiency of the program lowers greatly. Also, with three classes of images that contain no vehicle, and remain a dark or bright background, the program requires three kinds of different training models. Efficiency could be optimized through a simpler implementation created by Xie et al. [19]. With YOLO – a state-of-the-art object detection system based on a single neural network, we referenced and simplified some ideas from [19], and increased some efficiency while remaining the accuracy scale. Insightful ideas such as implementing character-segmentation-based plate detection [13], also have great potential for improvements.

For further implementation, license plate recognition could also be optimized into a real-time detection system. As Giannoukos et al. presented [9], the same procedures repeat in each frame of the video captured so that the program will be able to find related features among frames and extract number plates. However, since live videos contain more content than images, a real-time detection system should not only detect number plate features but also have functions to detect vehicle features in order to increase the total accuracy.

Character segmentation and recognition are usually considered separate processes. The current segmentation process has various approaches to achieve. A binary scale image processing is generally proposed [5, 6, 13, 15] to use add-up columns and rows which evaluate horizon and vertical vectors to detect regions that are valid for character segmentation. Simple Artificial Neural Networks (ANN) are common ways to train a character recognition model [14]. While this method is very efficient as it simply calculates possibilities with binary pixels, its accuracy on cropping characters is not always promising. Therefore, another approach worth mentioning is the gray-level thresholding and transformations introduced by [5]. Such an approach calculates a threshold that is extracted through a constant c from the mean gray level in an $m \times n$ window centered in the pixel. As shown in Figure 2, with an improved adaptive threshold from Lee et al. [12], there would be a .05 improvement in the binary scale image mentioned previously.

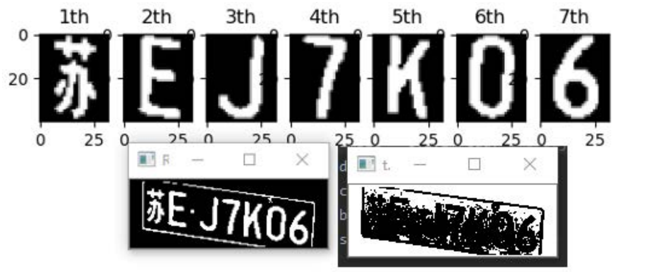

Figure 2: General Binarization Image (Bottom Right) and with An Adaptive Threshold (Bottom Left)

Character recognition systems had become very mature and effective at current days. It is evident that one of the most efficient ways to accomplish such functions is using feed-forward neural networks (ANN) [2]. According to reports made by Anagnostopoulos et al. [5], some accuracy performances from differently structured feed-forward neural networks along with their output class structures are shown in Table 1. An accuracy performance in such programs is defined by true positive, false positive, and false negatives, which in terms can be expressed as (1), in which we define a True Positive as features that are correctly identified, and False Negative as features that are incorrectly identified or failed to identify at all.

Table 1: Multilayered Feed-forward Neural Networks for Character Recognition [5]

Output Class Structure	Topology	Performance
9 digits	16-20-9	95%
26 English letters + 10 digits	24-15-36	98.5%
26 English letters + 10 digits	209-104-36	95%
8 digits of binary ASCII code	15-10-8	96%
25 letters + 10 digits	108-50-35	95%
26 letters + 10 digits	50-60-100-36	92.5%
36 alphanumeric classes	420-26-38	98.2%

$$Performance = \frac{TruePositive}{TruePositive + FalseNegative} \tag{1}$$

This paper [5] also presents the character recognition process with a feed-forward ANN, which implements 24 input, 15 hidden, and 36 output neuron layers. However, please notice that the difference is that no ANN is implemented into segmentation so far. In our following content, we will discuss how to enforce such a network into character segmentation in order to improve its performance.

In general, the biggest bottleneck for license plate recognition systems so far remains in optimizing accuracy within license plate detection. Compared with traditional image processing such as [1], our YOLO-based license plate detection reduced the average proceeding time to half of the original and optimized a total detection accuracy from 64% to 90.2%.

3 IMPLEMENTATION OF LICENSE PLATE DETECTION SYSTEM

3.1 Evaluate and Artificially Generate Training Dataset

As mentioned above, while most optimizations are implemented after evaluating the program's performance of certain datasets, we prioritize analyzing our dataset's noise and design a CNN-based training program depending on the analysis. Moreover, we generated 3,000 fake license plate boxes according to our noise analysis, as the current dataset cannot provide enough information for our training model to reach satisfying accuracy. Figure 3 shows our dataset, which contains up to 500 images taken under different times and weathers. There are also images whose license plates show as tilted. We analyzed potential banding noises generated from the camera within our dataset accordingly and decided to implement a Wiener filter [17]. A Wiener filter is a linear filter that can process images that are degraded by degenerate functions and contaminated by noise. The filtering method is based on that the image and noise are both random variables. It fits the condition well considering the variance within all the images. The Fourier domain of the Wiener filter presents as (2).

$$F(u,v) = [\frac{H*(u,v)S_f(u,v)}{S_f(u,v)|H(u,v)|^2 + S_\eta(u,v)}]G(u,v) \tag{2}$$

In the equation, H(u, v) represents a degenerate function, where H*(u, v) represents its complex conjugate; $|H(u,v)|^2 = H(u,v)H*(u,v)$; $S_\eta(u,v)$ represents image's original noise power spectrum, and $S_f(u,v)$ represents the power spectrum of ungraded image; G(u,v) is the Fourier transform of the degraded image.

In order to realize the noise-reducing function, and further implement our detection neural network for training, we artificially created the training dataset with SUN database [18]. SUN database is to provide researchers in computer vision, human perception, cognition, neuroscience, etc., with a comprehensive collection of annotated images covering a large variety of environmental scenes, places, and objects within. We generate 3,000 random license plate boxes with 8 strings, which are composed of 26 letters, 10 digital numbers, and 1 space. The plates can be tilted up to 45 degrees, and each training image has highly simulated textures as in our original testing dataset. Figure 4 shows an overlook of our generated training dataset.

3.2 License Plate Detection Neural Network Based on YOLO

The License Plate Detection program is one of the trickiest parts of the whole project. Simple image processing functions cannot properly handle image sets with a large number of different variables. Inspired by the Multi-Directional YOLO neural network [16, 19], we created the neural network for license plate feature detection, and train the program with our 3,000 generated images. YOLO is a real-time object detection algorithm that is both very efficient and accurate. Other than usual works on image detection, which is based on predicted high potential regions, YOLO uses a single neural network to the full image and separate images into regions. With such an approach, the efficiency improves greatly than traversing the entire image and predicting possibilities for each region. Our program proceeded on an RTX 2080 GPU, the procedure shown in Figure 5. With deep learning virtual machines provided by Google [7] and data set provided from COCO train 2014, 2080Ti can perform 81 images per second, with 32×2 batch size. The testing data contains up to 500 images with license plate features. The general test result from the training model reaches up to .89. More details should be revealed in Section 4.

3.3 Character Segmentation and Recognition

After detecting the license plate out of an image, the license plate should be cropped and extracted as a new output so that the next program can process with characters. The final step of the project, the Character Recognition program, would require characters to be separated and divided. Any image with more than one character would be identified as noise and would cause errors. Although we used a more efficient method in our license plate detection program, we were inspired by the approach which was generally used to traverse through images and look for high-scored regions and simulate the procedure in our character detection program. As Figure 6 shows, the orange box represents the region that the machine is processing. Each time the program will process a specific region out of the image, and look for potential patterns of any character. For each region, the program calculates its probability of character features. Finally, the program will list all the regions which represent character features with a high score, and crop those regions from the original image. This idea requires a trained model of character recognition. As we discussed in Section 2, models of character

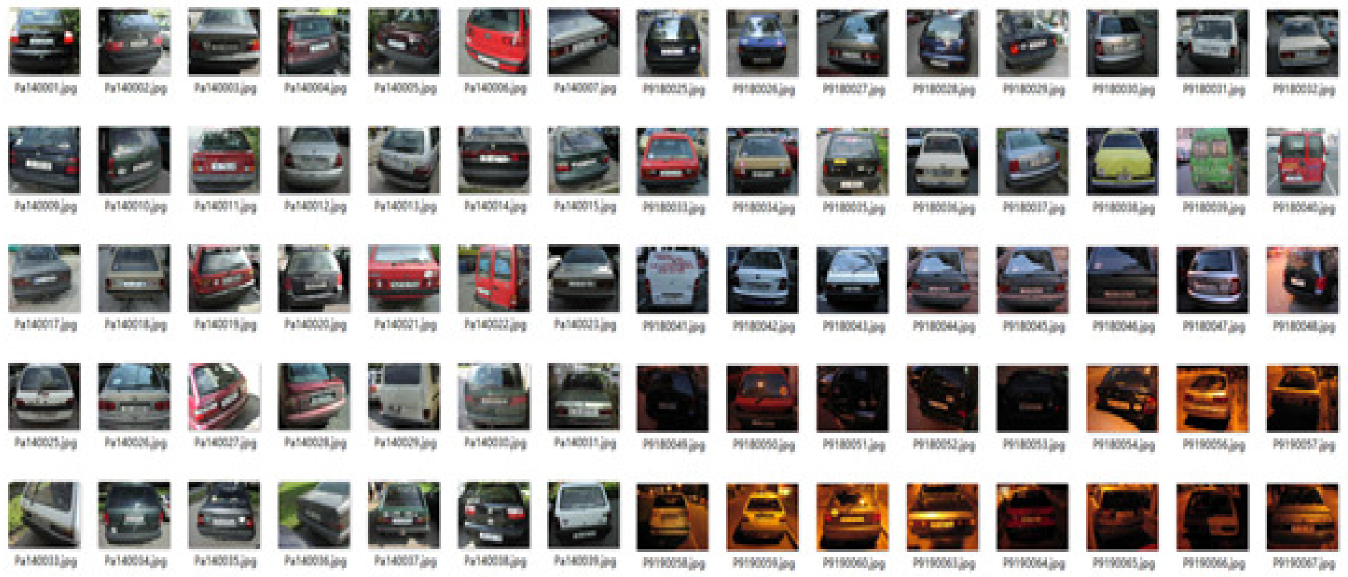

Figure 3: Sample of Images within The Dataset

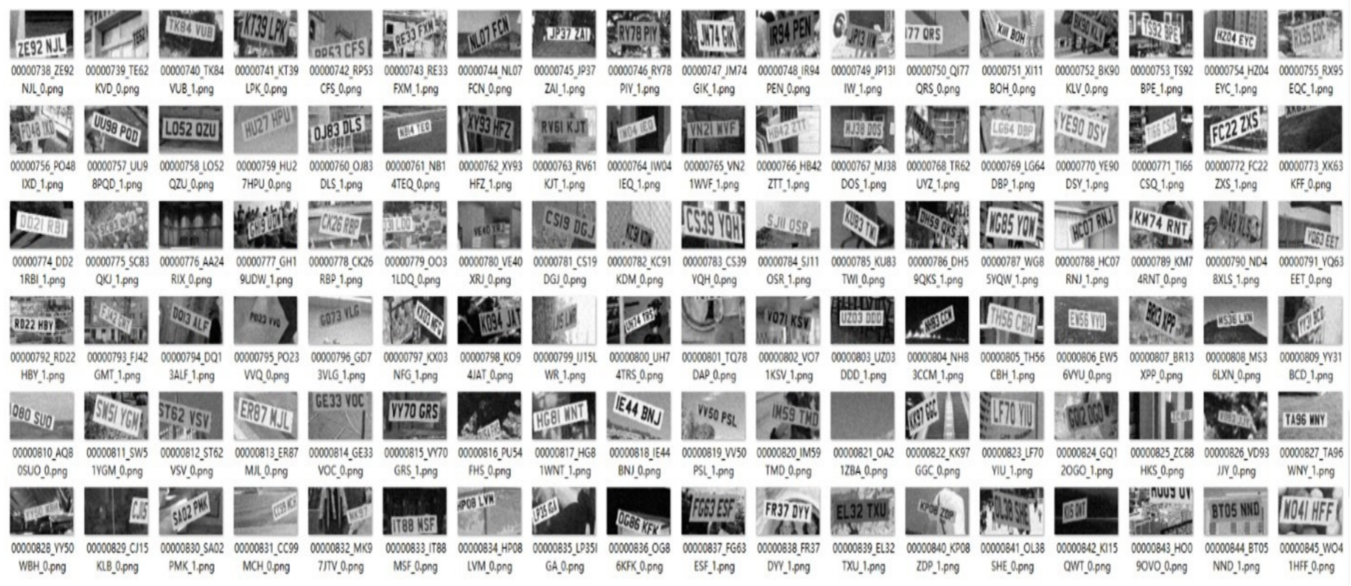

Figure 4: Gray-scale Images Contains A Random License Plate with Eight Total Strings

recognition are now very mature. However, there is another detail that is worth discussing. Some other character training programs [13] would use handwriting character datasets for license plate recognition, expecting a higher accuracy when doing the character recognition process. Considering the fact that most license plates contain well-printed characters only, some abstract handwriting forms of characters could in contrast affect recognition performance negatively. Therefore, we suggest considering certain textures of license plate characters and filtering out data that is not highly similar to printed styles. The dataset we used to train the program is the MNIST dataset [11], which includes a training set of 60,000 examples, and a test set of 10,000 examples. As Figure 7 shows, we extracted 900 data from each English and digital number and filtered out about 400 well-written data for each character.

Figure 8 shows a general procedure of our character segmentation and recognition program, the structure of character recognition is basically a simple ANN, which applies a 24-15-36 topology as [5] referred. Because recognize characters with some uncertain textures within our images, the accuracy of our character recognition program reached about 95.9%, which is a bit lower than Brugge et

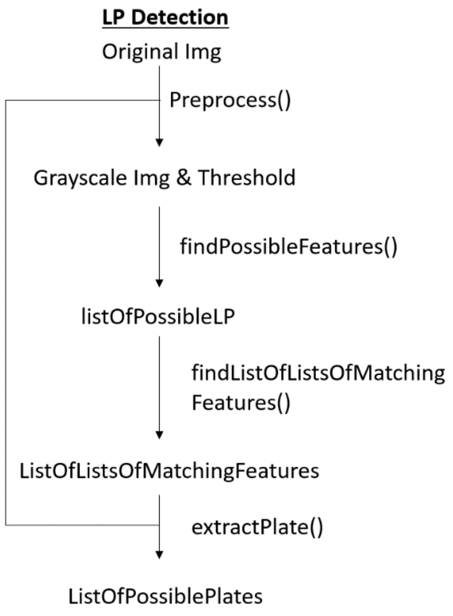

Figure 5: YOLO-based License Plate Feature Detection

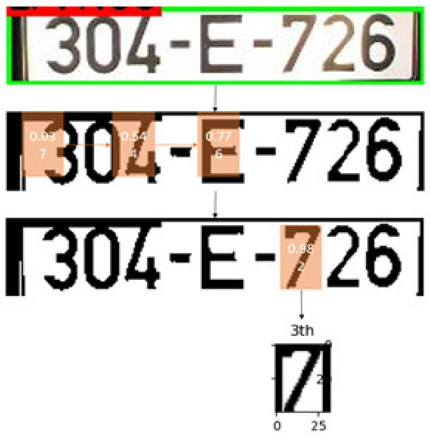

Figure 6: Character Segmentation

al. [5] originally reported. And because we cover both the character segmentation and recognition system together, the efficiency of the combined program did not decrease even though we use the region-by-region approach in order to increase our accuracy.

4 EXPERIMENTS AND EVALUATIONS

4.1 Experiment Setting

The entire experiment was done in a computer with an RTX 2080 graphic card and Intel 8-Core i7-8700K CPU. we generally tested 500 images from our dataset. In order to better featured each image's noise, we categorized all images into 10 groups, each group contains images with different backgrounds, such as rainy weather,

Figure 7: Samples of Character Recognition for License Plates

tilted license plate, nighttime image, etc. Table 2 shows detailed information about each group that we featured. In groups 4 and 5, license plates within images remain a light (5-15 degrees) or heavy (15-40 degrees) rotation; in group 9, some images have high contrasts, which means the color scale in the image is not obvious, and similar colors in clear pictures would be converted into a same scale. Before testing, we trained the program with our artificial training

Table 2: Group Features of Ten Groups

Data Group	Group Feature
1	50 Clear Pics
2	30 Clear Pics + 20 Pics with light fog
3	10 Clear Pics + 40 Pics with rainy weather
4	15 Clear Pics + 35 lightly tilted Pics
5	20 Lightly tilted Pics + 30 heavily tilted Pics
6	15 Clear Pics + 35 Low quality Pics
7	20 Clear Pics + 30 dark background Pics
8	5 Clear Pics + 45 dark background Pics
9	15 Clear Pics + 35 high contrast Pics
10	10 Clear Pics + 40 high contrast Pics

dataset with 30 batch-size and 100 epochs, the model would update parameters each time it activates the learning algorithm depending on its errors. In our testing process, Figure 9 shows some output results with the YOLO-based method, and the detection is robust as it can output most license plates that contain dark or bright textures.

4.2 Comparing License Plate Detection Algorithm with Traditional Image Processing Methods

We compared the YOLO-based algorithm with a traditional image processing and CNN algorithm proposed by Smara et al. [1]. With the same dataset, we trained the program and implemented a test, and record the performance of each algorithm. Tables 3 and 4 show the efficiency and accuracy of both algorithms. Figure 10 shows

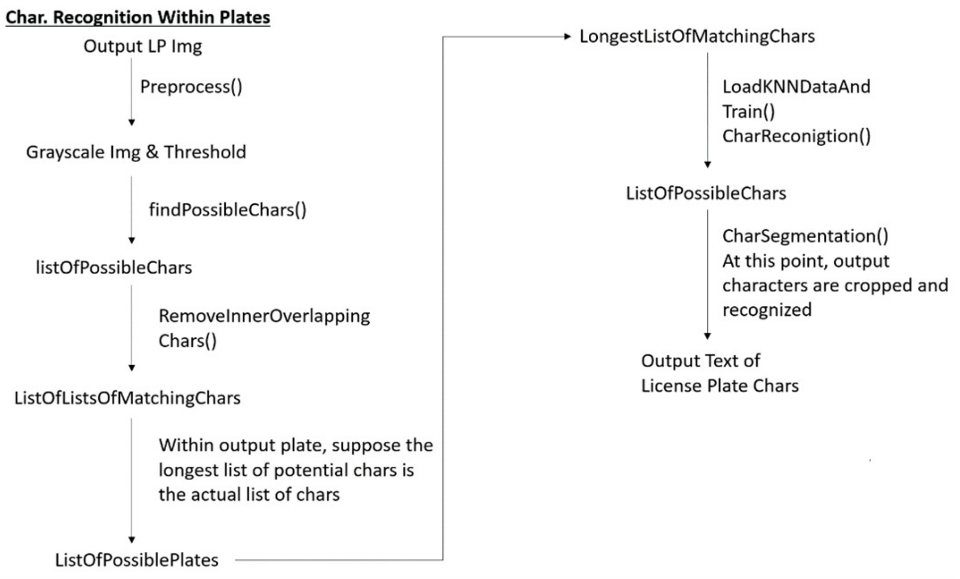
Char. Recognition Within Plates
Output LP Img
Preprocess()
Grayscale Img & Threshold
findPossibleChars()
listOfPossibleChars
RemoveInnerOverlapping Chars()
ListOfListsOfMatchingChars
Within output plate, suppose the longest list of potential chars is the actual list of chars
ListOfPossiblePlates
LongestListOfMatchingChars
LoadKNNDataAnd Train() CharReconigtion()
ListOfPossibleChars
CharSegmentation() At this point, output characters are cropped and recognized
Output Text of License Plate Chars

Figure 8: Basic Process of Character Segmentation and Recognition Program

the two algorithms' regional and total performance with the entire testing dataset.

Table 3: Comprehensive Performance of YOLO-based License Plate Detection Program

Data Group	Time	Regional Accuracy	Total Accuracy
1	208.73 sec	90%	90%
2	229.42 sec	98%	94%
3	224.8 sec	94%	94%
4	220.45 sec	88%	92.5%
5	243.06 sec	90%	92%
6	202.93 sec	92%	92%
7	209.06 sec	86%	91%
8	189.45 sec	84%	90.3%
9	214.56 sec	88%	90%
10	200.39 sec	92%	90.2%
Total	2442.85 sec	-	90.2%

Please note that all performances are evaluated based on equation 1. Regional performance represents the accuracy that each algorithm performed within each featured group, and total performance represents the current total accuracy combined with all former regional accuracy. We evaluate our program with the performances of each group, as different groups contain similar textures so that we could easily see whether the program is good or not at dealing with certain types of images. As the report shows, we receive a total of 90.2% accuracy with an average time required for each image is about 4 seconds. Compared to the traditional image process and CNN algorithm, where the total accuracy reaches about 63%, and the time spent for each image is about 15.27 sec, the efficiency of the YOLO-based algorithm is considered good. Most groups received an accuracy above 90%, while some others such as Groups 8 and 9, these groups contain images with huge noises, such as foggy weather, extremely dark backgrounds, high image contrast, and so on.

Table 4: Comprehensive Performance of Image Process and CNN License Plate Detection Program

Data Group	Time	Regional Accuracy	Total Accuracy
1	530.5 sec	68%	68%
2	547.2 sec	74%	81%
3	537.5 sec	68%	70%
4	525.8 sec	70%	70%
5	492.2 sec	60%	69%
6	522.2 sec	64%	68%
7	515.7 sec	62%	67%
8	398.8 sec	52%	66%
9	442 sec	58%	65%
10	452.5 sec	60%	64%

4.3 Evaluation of ANN Character Recognition

Our character segmentation and recognition program are based on output data of the license plate detection's performance. In our experiment, we totally received 451 cropped license plates out of 500 data. We also classify all output data with the same groups we used previously, as the cropped license plates contain identical textures to their original images. Table 5 and Figure 11 show our total performance of the character segmentation and recognition program. As the character recognition program can only process with image output by a former license plate detection program, we evaluate its performance differently. Instead of calculating the accuracy of each group, we set the number of each group's successfully detected license plates as the input of character recognition.

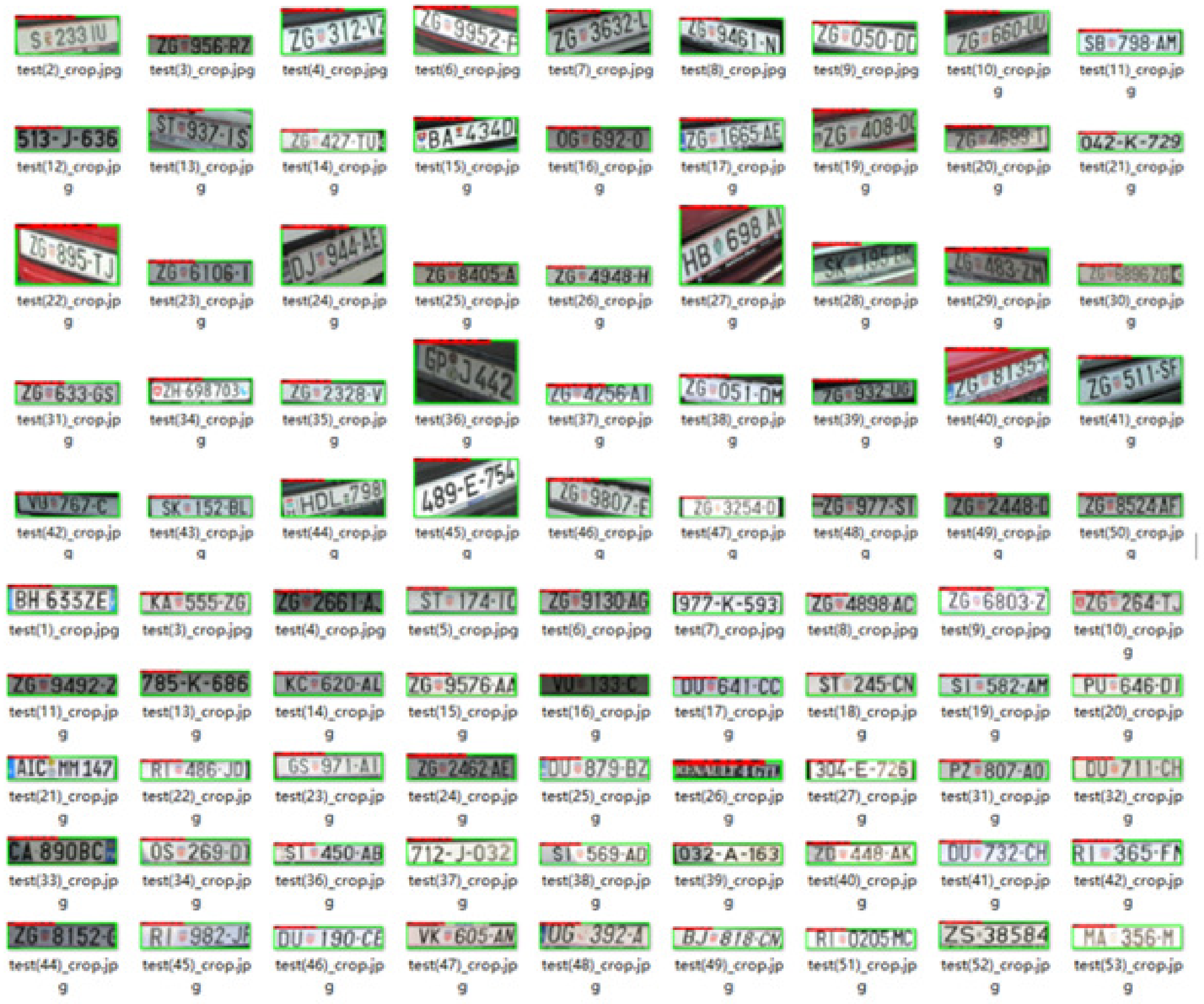

Figure 9: Samples of Detected License Plates

Originally presented by Anagnostopoulos et al. [5], although we did not reach a 98.5% accuracy in our progress, as the data which our program processed contained more textures than a binary dataset for character recognition, a total accuracy of 95.9% also proved that such character recognition system is robust. However, the same issue remains as in our license plate detection program; for example, in Group 8, in which the images are mostly taken at nighttime and contain very dark environments, the performance of both license plate detection and character recognition programs remain lower than most other groups. Therefore, we suppose that image processing algorithms might require additional training for dark contexts. Bulan et al. [3]'s approach of using different datasets to train dark and bright classifiers also support this assumption.

5 CONCLUSION

In this paper, we discussed the potential improvement of designing algorithms with prioritized dataset analysis and proposed a simplified YOLO-based CNN approach for license plate detection. With a specified optimization of our dataset, the program could improve its efficiency while still reaching a 90.2% performance accuracy. Also, the entire character segmentation and recognition programs can be simplified by combining them together. The approach of implementing the ANN system to segmentation program also improves its performance and reaches a satisfying result. However, as we discussed in Section 4, there are still bottlenecks remaining in our recent works. It is not easy for our programs to deal with dark images without specific training for a certain texture. Also, as such images could possibly contain much more complex noises

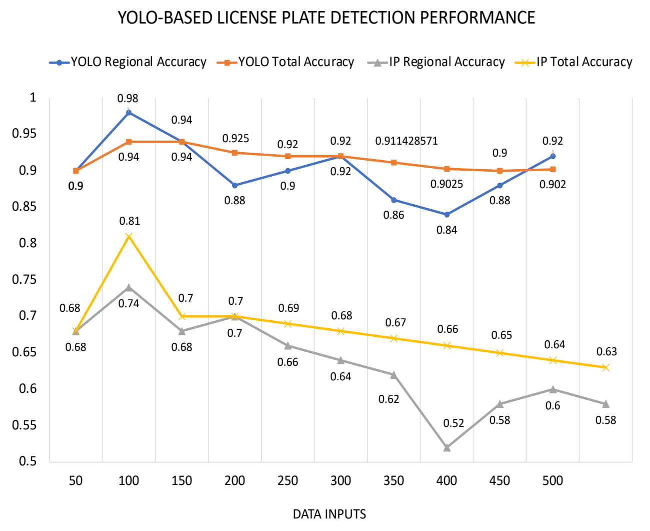

Figure 10: Regional and Total Accuracy Comparison of YOLO-based and CNN-based algorithms

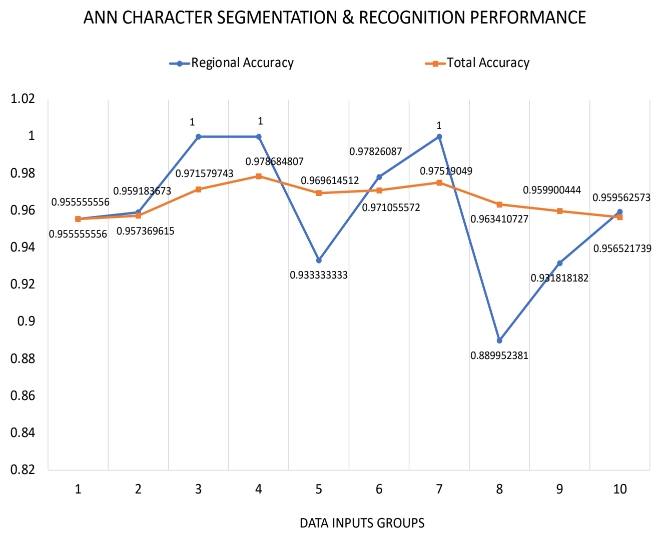

Figure 11: Regional and Total Accuracy of ANN Character Segmentation and Recognition

Table 5: Multilayered Feed-forward Neural Networks for Character Recognition [5]

Data Group	Regional Accuracy	Total Accuracy
1	0.955555556	0.955555556
2	0.959183673	0.957369615
3	1	0.971579743
4	1	0.978684807
5	0.933333333	0.969614512
6	0.97826087	0.971055572
7	1	0.97519049
8	0.880952381	0.963410727
9	0.931818182	0.959900444
10	0.956521739	0.959562573

and some features may appear undetectable, methods of processing such images might be beyond image feature detection.

ACKNOWLEDGEMENTS

This research was supported in part by US NSF grants CNS-2244450, CNS-2103405, AMPS-2229073, CPS-2103459, and SHF-2210744.

REFERENCES

[1] Samra G Abo and Khalefah Farr. 2013. Localization of License Plate Number using Dynamic Image Processing Techniques and Genetic Algorithms. *IEEE transactions on evolutionary computation* 18, 2 (2013), 244–257.
[2] Elena Baralis, Luca Cagliero, Tania Cerquitelli, Vincenzo D'Elia, and Paolo Garza. 2010. Support Driven Opportunistic Aggregation for Generalized Itemset Extraction. In *International Conference Intelligent Systems*. IEEE, London, UK, 102–107.
[3] Orhan Bulan, Vladimir Kozitsky, Palghat Ramesh, and Matthew Shreve. 2017. Segmentation-and Annotation-free License Plate Recognition with Deep Localization and Failure Identification. *IEEE Transactions on Intelligent Transportation Systems* 18, 9 (2017), 2351–2363.
[4] Anagnostopoulos Christos-Nikolaos, Anagnostopoulos Ioannis, Psoroulas Ioannis, Loumos Vassili, and Kayafas Eleftherios. 2006. A License Plate-recognition Algorithm for Intelligent Transportation System Applications. *IEEE Transactions on Intelligent transportation systems* 7, 3 (2006), 377–392.
[5] Anagnostopoulos Christos-Nikolaos, Anagnostopoulos Ioannis, Psoroulas Ioannis, Loumos Vassili, and Kayafas Eleftherios. 2008. License Plate Recognition from Still Images and Video Sequences: A Survey. *IEEE Transactions on intelligent transportation systems* 9, 3 (2008), 377–391.
[6] Duan Tran Duc, Du Hong, Phuoc Tran Vinh, and Hoang Nguyen Viet. 2005. Building An Automatic Vehicle License Plate Recognition System. In *Proc. Int. Conf. Comput. Sci. RIVF*, Vol. 1. Citeseer, Can Tho, Vietnam, 59–63.
[7] Google. 2006. Deep Learning VM Image; Google Cloud. https://cloud.google.com/deep-learning-vm/
[8] Selena He, Meng Han, Jack Zheng, and Herman Ray. 2020. Implementing Capsule Neural Networks in Traffic Light Image Recognition. In *2020 ACM South East Conference*. ACM, Tampa, FL, USA, 301–302.
[9] Giannoukos Ioannis, Anagnostopoulos Christos-Nikolaos, Loumos Vassili, and Kayafas Eleftherios. 2010. Operator Context Scanning to Support High Segmentation Rates for Real-Time License Plate Recognition. *Pattern Recognition* 43, 11 (2010), 3866–3878.
[10] Barroso James, Dagless EvaL, Rafael Arricus, and Bulas-Cruz Jullius. 1997. Number Plate Reading using Computer Vision. In *ISIE'97 Proceeding of the IEEE International Symposium on Industrial Electronics*. IEEE, Guimaraes, Portugal, 761–766.
[11] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. 2006. The MNIST Database. http://yann.lecun.com/exdb/mnist/.
[12] Byeong Rae Lee, Kyungsoo Park, Hyunchul Kang, Haksoo Kim, and Chungkyue Kim. 2004. Adaptive Local Binarization Method for Recognition of Vehicle License Plates. In *International Workshop on Combinatorial Image Analysis*. Springer, Guimaraes, Portugal, 646–655.
[13] Hui Li and Chunhua Shen. 2016. Reading Car License Plates using Deep Convolutional Neural Networks and LSTMs. *arXiv preprint arXiv:1601.05610* (2016).
[14] Chirag Patel, Dipti Shah, and Atul Patel. 2013. Automatic Number Plate Recognition System (ANPR): A Survey. *International Journal of Computer Applications* 69, 9 (2013).
[15] Juauedur Rahman, Suaur Beauchemin, and Michael Anthony Bauer. 2019. License Plate Detection and Recognition: An Empirical Study. In *Science and Information Conference*. Springer, 339–349.
[16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-time Object Detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. Las Vegas, USA, 779–788.
[17] Kumar VR Vijay, Manikandan S, Ebenezer D, Vanathi PT, and Kanagasabapathy P. 2007. High-density Impulse Noise Removal in Color Images using Median Controlled Adaptive Recursive Weighted Median Filter. *IAENG International journal of computer science* 34, 1 (2007).
[18] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. 2010. Sun Database: Large-scale Scene Recognition from Abbey to Zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE, San Francisco, USA, 3485–3492.
[19] Lele Xie, Tasweer Ahmad, Lianwen Jin, Yuliang Liu, and Sheng Zhang. 2018. A New CNN-based Method for Multi-directional Car License Plate Detection. *IEEE Transactions on Intelligent Transportation Systems* 19, 2 (2018), 507–517.