Detection Of Vehicles and The License Plate Using Convolutional Neural Networks (CNN)

كشف السيارات ولوحات الترخيص باستخدام الشبكات العصبية الالتفافية (CNN)

Ali Kadhim Mohammed
Department Of Computer Science, University of Diyala
الباحث علي كاظم محمد قسم علوم الحاسبات /جامعة ديلى
Email: almalikiali771@gmail.com

تاريخ تقديم البحث: 2024/02/10
تاريخ قبول النشر: 2024/02/28

Abstract

For this research, Convolutional Neural Networks (CNN) were used to identify and recognize automobiles and license plates. This study has focused on training neural networks with plenty of data. Building on my previous research on neural networks and neurons, I want to be able to identify and recognize cars and their license plates. The research will begin with neural network basics. Both CPU and GPU processing power will be used to train neural networks. Several GPUs, the CUDA kernel, and the keras library have been used for matrix backpropagation. OCR and object identification are the key technologies utilized to address this issue. Many image preparation methods are used to optimize image clarity within parameters. Most of these methods distort images by skewing and darkening them. Darkflow was used to train a convolutional neural network for You Only Look Once object identification. In our examination, 99.7% of tags of the necessary quality were correctly read with our most current item. Without picture preprocessing, the proportion plummeted to 6.95%. I used Keras and tensor flow environments from Anaconda for Python to construct algorithms for car identification, photo registration, and YOLO object detection and recognition using deep learning and OpenCV.

Keywords : Detection of Vehicles; CNN: YOLO: Darkflow.

الملخص

في هذا البحث، تم استخدام الشبكات العصبية الالتفافية (CNN) لتحديد والتعرف على السيارات ولوحات الترخيص. ركزت هذه الدراسة على تدريب الشبكات العصبية باستخدام الكثير من البيانات. استنادًا إلى أبحاثي السابقة حول الشبكات العصبية والخلايا العصبية، أهدف إلى القدرة على تحديد والتعرف على السيارات ولوحات ترخيصها. سيبدأ البحث بأساسيات الشبكات العصبية. سيتم استخدام قدرات المعالجة لكل من وحدة المعالجة المركزية (CPU) ووحدة معالجة الرسومات (GPU) لتدريب الشبكات العصبية. تم استخدام عدة وحدات معالجة رسومات ونواة CUDA ومكتبة keras للارتداد العكسي للمصفوفات. تُعد تقنيات التعرف الضوئي على الأحرف (OCR) وتحديد الكائنات هي التقنيات الرئيسية المستخدمة لمعالجة هذه المشكلة. تم استخدام العديد من طرق تحضير الصور لتحسين وضوح الصورة ضمن المعايير. غالبية هذه الطرق تشوه الصور عن طريق الإمالة والتعتيم. تم استخدام Darkflowلتدريب شبكة عصبية التفافية لتحديد الكائنات باستخدام تقنية .You Only Look Once (YOLO) في فحصنا، تم قراءة 95.3٪ من العلامات بالجودة المطلوبة بشكل صحيح مع أحدث عنصر لدينا. بدون تحضير مسبق للصور، انخفضت النسبة إلى 6.95٪. استخدمت بيئات keras و tensor flowمن anaconda لـ Python لبناء خوارزميات لتحديد السيارات، تسجيل الصور، وتحديد الكائنات والتعرف عليها باستخدام تقنية YOLO والتعلم العميق و.OpenCV

1. Introduction

Machine learning, artificial intelligence, and neural networks have become increasingly popular [1]. It lets computers autonomously examine enormous volumes of data for patterns. The data used may be too large to analyze, understand, and draw conclusions from, making it beneficial. With the capacity to drive driverless cars and make the perfect pint, this technology has many applications. Computer vision has significantly benefited from the fast rise of machine learning. After training, computers can discover similarities between newly generated photos and patterns in older photos, eliminating the need for human interpretation [2]. Computer vision software can also analyze images and transform alphanumeric characters into text. Our initial expectations were met by project completion. Our optical character recognition (OCR) service can properly process several license plates.

Additionally, our system can identify and isolate license plates. Despite our program's underperformance, real-world phone testing showed that our initial estimates were too optimistic [3]. Electricity is essential in modern society. Power pylons, transformers, and circuit breakers make up the worldwide power supply network. Keeping power flowing requires regular infrastructure repair. The attached license plates provide vital administrative and maintenance data [4]. Vendor names, serial numbers, and manufacturing years are often printed on license plates. Logos, schematics, and barcodes may be present. License plates are rectangular and made of metal or plastic. Translating text graphics into machine-readable language allows access to license plate semantic data. It may be called optical character recognition. OCR software and other ways have been used to overcome this issue [5]. Most modern systems are designed to handle scanned paper sheets.

Additionally, each of the four license plates uses a typeface different from text processing typefaces, adding to the intricacy. The plate with embossed lettering has less contrast and stroke width than the others because it was neither painted nor printed. Thus, present optical character recognition technologies may yield unsatisfactory results. Our approach to recognizing, classifying, and extracting license plate text is novel. These license plates must be rectangular and have legible text. They are for power supply equipment mounting—no analysis or interpretation of schematics or barcodes. Our process includes these steps: After recognizing the license plate in the picture, delete and rotate it clockwise. Machine learning was used to extract attributes to determine the category [6]. Each category should include a complete list of text segments to extract and their placements. Regions are preprocessed to remove particles, neighboring text, and text frames. An optical character recognition engine receives the targeted areas. If a spot's identification score goes below a threshold, the user is instructed to take a new photograph of the affected plate region. After comparing the new image to the old one, OCR is performed. The new image may only cover part of the license plate. Our multi-view technique ensures accurate text extraction, even with reflections or occlusions. According to reference [7], the algorithm's final output comprises plate type identification, text recognition, text arrangement, and confidence in the findings. A functioning prototype that retrieves photographs and extracts pertinent information is the project's primary goal.

The exam focuses on classifying license plates assessing each aspect separately and jointly. It also shows that training the OCR system to recognize plate fonts improves its performance. Finally, it shows how illumination affects text extraction and plate classification. This [8] includes numerous key failure reasons. In addition to multiple sample plates, OMICRON Electronics GmbH submitted several photos of license plates from transformers and related equipment [9].

The problem statement for detecting vehicles and license plates using Convolutional Neural Networks (CNN) involves designing and training a model that can accurately identify vehicles in images or video streams, extract their license plates, and recognize the characters on those plates. This task typically requires a large dataset of labeled vehicle images with corresponding license plate annotations, as well as expertise in designing CNN architectures suitable for object detection and character recognition tasks. The goal is to develop a robust and efficient system capable of accurately detecting and recognizing vehicles and their license plates in various environmental conditions and camera perspectives.

1.1. Contribution

While publicly accessible datasets were employed for detection in this piece, the primary aim should have been to identify the license plate precisely for potential utilization in my subsequent work on license plate recognition. The second purpose is to demonstrate the proficiency with which this study venture will handle the license images to the audience. Therefore, the suggested system must show the vehicle's license plate in both alphanumeric formats while also operating inconspicuously. An optical character recognition service will have a higher level of comprehension with a high-resolution image. The utilization of our company's data in conjunction with the YOLO approach is recommended for the training of a CNN to achieve object recognition in our application and employ the YOLO algorithm to get optimal performance. This study employs the OpenCV library along with deep learning libraries to develop techniques for vehicle identification and picture reconstruction using YOLO object detection and recognition. Additionally, I utilize Python libraries such as Keras, TensorFlow, and the Anaconda environment for Python programming. My study will leverage several object tracking benchmarks, such as CNN and LTSM-Bounding Boxes, along with the Anaconda Navigator platform.

2. Methodology

This section provided a comprehensive explanation of the methodology employed to identify and recognize automobiles and license plates through the utilization of advanced deep-learning algorithms. The procedure of extracting the required information from a given set of license plates is explained, with the condition that only text that humans can easily read is considered. This study excludes the examination of barcodes and schematic drawings. Figure 1 displays a schematic representation of the planned procedure. The initial stage involves locating the license plate, detaching it, and subsequently altering its appearance. The kind of plate may be determined by the resulting picture, which exclusively displays the license plate in a vertical orientation. As a result of the absence of uniformity in this sector, manufacturers generate a wide variety of designs for their ranges of license plates. The classifier should possess the capability to acquire knowledge swiftly to accommodate the inclusion of novel plate types as soon as possible, given the extensive range of plates now in circulation. Random Forests, known for their efficient classification capabilities, are commonly employed for this objective. They exhibit excellent performance on mobile devices. To get the necessary information for classification, the image is partitioned into cells by overlaying a grid. Generate a feature vector by utilizing the median color value and the histograms of Local Binary Patterns (LBP) for each grid cell. Finally, incorporate the plate's size ratio into the total cell values. As part of our operations, i have a list that contains the specific designations and places of the text sections that need to be obtained. This list is based on a collection of recognized license plates for each category.

Given that plates of a specific kind have identical patterns, identifying the locations of text sections simplifies the process of extracting content by eliminating the need to create field boundaries. The capacity to "cartographically represent" data from one location to another using labels is also accessible; this is essential for comprehending the data in the future. The gathered regions undergo preprocessing to eliminate debris and other structural imperfections on the plates. Preprocessing involves the grouping of characters, filtering based on size, and applying morphological opening and closure operations. To complete the process, the preprocessed text areas should be transmitted to the Tesseract OCR engine. If the recognition score of a particular spot drops below a designated threshold, the user is prompted to capture a fresh image of the afflicted area on the plate. The updated photos are consistent with the present as shown in Figure 1.

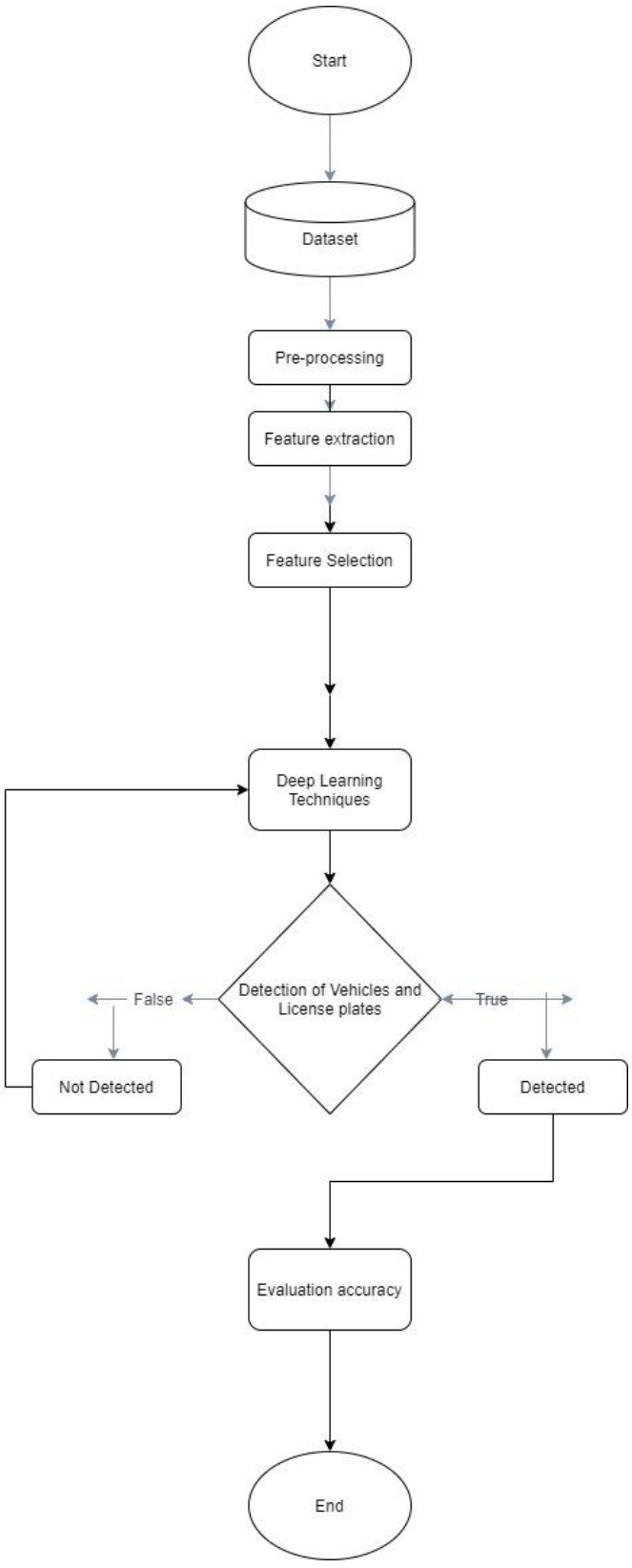

Fig. 1: A diagram illustrating the process of identifying automobiles and registration plates using Convolutional Neural Networks.

2.1 Ocr Service

Upon receiving the cropped image, the service anticipates a depiction where the license plate dominates the majority of the image. The image requires enhancement to ensure legibility and effectively convey the essential information, namely the characters. The first phase of the program is to apply filters to enhance the visibility of the plate, which will subsequently allow for the synchronization of its actual borders. Subsequently, the application should apply a blurring technique to the image in order to eradicate any noise, scratches, and superfluous information. In order to improve its clarity, the image has undergone the processes of binarization, grayscale conversion, and maybe distortion. The application will distort the license plate to conform to the dimensions of the picture and eliminate any bothersome elements that are not part of the plate. Once the photo has undergone preprocessing, it should be forwarded to the subsequent stage of the pipeline. The OCR service often delivers the extracted text from the cropped image in less than one second. Even in the most extreme and urgent situations, a duration of 10 seconds is inadequate. Subsequently, the application will utilize the OCR service to extract text from the picture. Ensuring the image is adequately prepared in the preceding stage will facilitate the OCR service's task and enhance its accuracy in generating the correct sequence of letters.

2.2 Dataset

Upon receiving the SMS, the database should promptly get the license plate data of the car and transmit it to the phone. Having this data available can streamline the process of submitting revised vehicle details, including photographs and reports on damages. Although the database is not included in the project scope, it is provided to provide you with a comprehensive understanding of how our system is intended to function.

2.3 Building an object detector.

Our main goal was to develop an exact object detector that could operate at maximum speed—initially, we intended to employ the YOLO methodology due to its rapidity. However, the implementation of the Darknet proved incompatible with smartphones. Nevertheless, TensorFlow is supported by a genuine open-source project that effectively utilizes TensorFlow. Fortunately, this project embraced the concept of YOLO. Darkflow was the optimal solution for this circumstance due to its YOLO model, which is compatible with TensorFlow.

2.4 Training

Prior to training our CNN model, it was necessary to preprocess the training data. We successfully trained a substantial volume of data, as anticipated. The final collection comprised over 3500 photographs of cars with different levels of visibility of license plates. Each image must be annotated, as Darkflow's CNN relies on supervised learning. To utilize object detection, it is imperative to manually delineate a square over each image in the dataset that one intends to recognize, followed by annotating it with the corresponding item. Figure 2 illustrates this procedure.

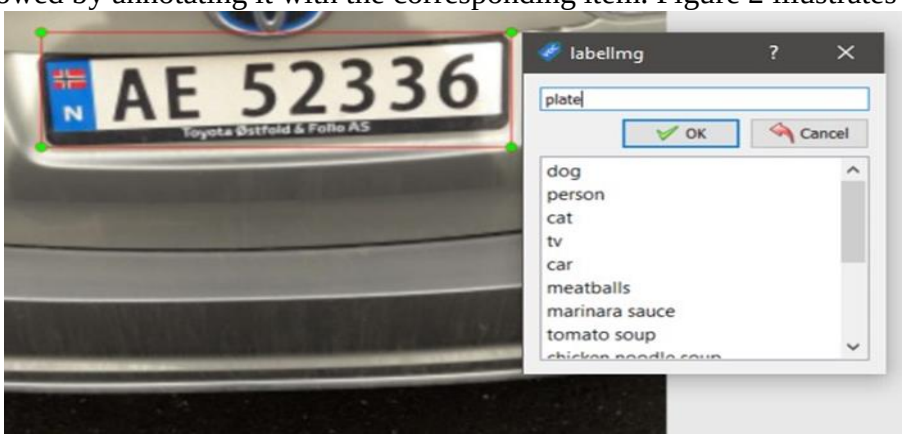

Fig. 2: Adding descriptive labels to an image of a vehicle's registration plate.

Prior to commencing a training session, it is necessary to obtain the configuration file and pre-trained weights. The configuration file is a textual document that provides a detailed description of the precise arrangement of the CNN. It establishes the overall quantity of convolutional layers and the specific tuning of each layer. Furthermore, it is essential to alter the configuration file to enable the object detector to accurately identify the number of items, precisely one in this scenario.
The collection has photographs of both automobiles and objects that are not vehicles. All the photos have dimensions of 64x64 pixels and are in the RGB color format.

- Vehicle Count: 3500
- Non Vehicle Count: 1500

Fig. 3: Sample Vehicle Image in Dataset

Initially as shown in Figure3 , they trained CNNs and subsequently utilized their pre-learned weights as a foundation. A conventional approach is to train a CNN by utilizing pre-existing weights on large datasets, which enables the CNN to acquire knowledge of essential characteristics such as edges and corners. Subsequently, the model is directed to identify specific objects, such as license plates. During our training, I often utilized the readily available pre-trained weights from the official Darknet page, specifically the Tiny YOLO weights. The model option designates the file to be utilized for configuration as shown in Figure 4.

Fig. 4: Sample Non-Vehicle Image in Dataset

The ultimate version of our configuration file, tiny-yolo-voc-graph. cfg encompassed all of our modifications. The load determines the selection of pre-trained CNNs. Due to our lack of familiarity with CNNs, we chose to utilize the pre-trained weights. Subsequently, when executed, the same CNN once more employed the load command to get the previously trained CNN. Following Trainer Adam's recommendation to utilize the Adam optimizer during training, Train

instructs Darkflow to begin the procedure. The dataset and annotations refer to the file directories that Darkflow uses to locate the images and annotations that have been previously prepared. Darkflow optimizes its performance by using GPU 0.75, which effectively utilizes 75% of the available VRAM and alleviates the CPU's workload.

The optimal setup of our model was crucial, as previously said. Our approach is quite iterative since we have trained 14 CNNs using different quantities and qualities of input and, during the training period, conducted weekly training sessions for two models, one on weekdays and another on weekends. The last stage involved evaluating the performance of the CNNs by comparing their results on two distinct datasets comprising photos that had not been encountered before. The speed of training had a crucial role in determining the quality of the final object detector, given the time constraints of this project. GPU training has become more crucial compared to CPU training due to the significantly higher training speed of GPUs in training neural networks, and a training session using the CPU was conducted to compare the two. The results demonstrated that GPU training was more than eight times faster, with a rate of 2,100 training steps per hour compared to 250 steps per hour for CPU training. The GPU utilized for this assignment was an NVIDIA Quadro M2000M, which may be described as relatively slow. In order to expedite the training process, we developed a Python script to rescale all of our photographs. I typically received oversized original pictures, measuring between 2000 and 3000 pixels in both width and height, which exceeded the necessary proportions—by utilizing our script, I maintained the aspect ratio while decreasing the maximum width and height to 500 pixels. Although no comparison was made, this approach accelerated the training process and decreased the image size by a factor of 10 to 20.

3. Results

used deep learning to create an object detector and OCR service in this session. Both components focused on license plates and vehicle detection. The object detector was built using TensorFlow and tweaked to meet our needs. Created a separate optical character recognition (OCR) service to interpret photos taken with the app.

This section evaluates our OCR service and object detection accuracy. Also, evaluate our photo processing.

As always, performance was our first goal while developing our object detector. Smartphone technology has improved, but real-time object detection is still challenging. Used to capture a picture, execute the YOLO technique, and outline a license plate bounding box in half a second.

This results in two updates per second. According to project discussions, the program should make 5-10 changes per second to ensure a flawless launch look. I verified it live and found it false. Even though the app only updated twice per second, the camera preview worked at its maximum frame rate, making it smooth. The program's parking lot application is essential. Maintaining camera stability prevents the camera from lagging when the box is rendered at precise screen coordinates, like during rotation. Pointed the camera at a plate, pushed the shutter button, and let object detection adjust for a while to inspect it. A green box was easy to get with 90% certainty when the camera was pointed squarely at a car's front or back. Photo acquisition and distribution were unimportant. The license plate is abbreviated, as shown in the camera preview. All processing stops when the shutter button is pressed, and the photo is sent to our OCR service in a separate thread without user involvement.

Ran the object detector on test photographs to evaluate the result. These photos were taken to test the object detector, not for optical character recognition, and a Python script was needed to recognize license plates with 50% certainty. The script used our model and YOLO to process all test photographs while monitoring each instance. Out of 147 plate-containing photos, 138 were discovered. The photos varied in quality and viewpoint. Amazingly, the training set of images had 94% accuracy. I used the same script on our original dataset of 3500 photographs but increased the

confidence level to 90% to test our OCR service. The software was modified to gather and save matching license plates. The goal was to gather photos shot using the software when the box turned green. This section evaluates our OCR service using 1026 images.

Since these photos were used to train the object detector, they may influence its judgment. I evaluated the OCR service by comparing its speed, accuracy, and durability. I discuss how to achieve consistent photo identification in the application while minimizing the harmful effects of inactivity and using just OCR software for image interpretation.

Analyze the benefits of picture preparation before analysis. When uploading static photographs that cannot be instantaneously rectified into an OCR service, reliability is crucial. A downside of dependable OCR services is performance. I could spend all day experimenting with photographs to comprehend them. I prefer not to do this since sending data to an application that must reply instantly is cumbersome. Therefore, picture preprocessing must be fair at every stage. Most smaller license plate photos are 150 pixels wide, making them noticeable and fast to load. Reading photographs is the hardest part of managing larger images. Thus, OCR software becomes slower than the OCR service while handling higher-resolution photos. The OCR software's attempts to assess its performance shall be counted. This lets us separate the two main components. Picture preprocessing and the reader comprise an OCR system. The minimum picture size OCR can read is 150 pixels wide or 5-10 kilobytes. Maximum picture size is 100-200 kilobytes with 1000 pixels width, or twenty times bigger. The hardware on which a service processes a photo determines its processing time.

Comparing photos of different sizes is better than focusing on-time performance. A database with all the license plate numbers will be analyzed, plus many more in a modifiable format, to evaluate the OCR service's license number recognition and prediction skills.

The database's numerical data will be compared to the OCR estimate. We will start by testing the data transfer speed of two standard images, 5 to 10 kilobytes and 100 to 200 kilobytes, respectively. This test verifies that the OCR service can adequately read the preprocessed picture in one try without iterations. Iteration and preprocessing are required when the OCR service estimate does not match license number pattern criteria, as illustrated in Figure 5. The larger image has 40 times the area and roughly 18 times the disk space of the smaller image, but it takes four times longer to preprocess and read. Comparing the photos shows this. Fortunately, analyzing more extensive photographs takes logarithmically longer. Only non-preprocessed data comparisons are affected as shown in Figure 5.

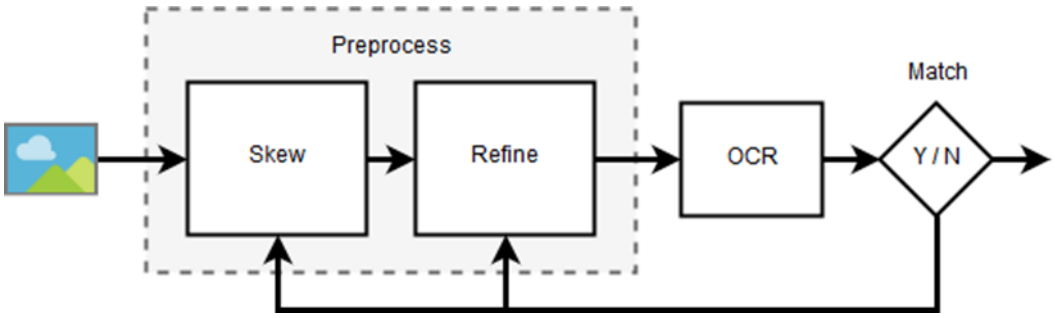

Fig. 5: The picture is processed by a flowchart in the OCR service pipeline. Initially, it undergoes preprocessing before being analyzed by the OCR software. Subsequently, it is compared to determine if there is a match or not.

The initial estimate is wrong; the OCR service has two iterations. This requires repeated skewing and refining. The initial iteration algorithm's success in finding an arbitrary quadrilateral is questionable. In the next round, the requirement will be tightened if the computer fails to find the quadrilateral. Thus, the software must adjust to skew. If this function fails to find an arbitrary quadrilateral, it will only iterate twice. The configuration alteration is sensitive and requires only little tweaks to enhance its tolerance when detecting a new quadrilateral. Therefore, restrict repeats

to two. Enhancing the visual usually follows skewing. The refining method assumes a suitably distorted picture. The threshold increases or decreases as the computer slowly adjusts the binarization input during each refining cycle. The value increases with time.

The OCR service then analyzes and infers picture information to find a new connection. However, additional iterations are improbable. The computer usually gives an exact or imperfect estimate after distortion and enhancement. Since this step of the service is the longest, the optical character recognition program should prevent multiple picture scans. This led us to ask essential questions, such as whether app users would rather wait for the app to respond, which provides more accurate information but may take 20 seconds than receive a notification that the car could not be found. What if the customer accepts a less accurate car estimate, but the algorithm still takes three seconds to find the right match instead of twenty.

Separating the three types of plate characteristics and assessing them separately is necessary to establish their importance. Following the grid search, compute the features using the optimum parameters. Then combine them and evaluate their contributions to classification.

A random forest with 1% accuracy, 50 trees deep, and 300 trees maximum may be trained. Finding the square root of the characteristic vector's number of characteristics gives the division's magnitude as shown in Figure 6, Figure 7 and Figure 8.

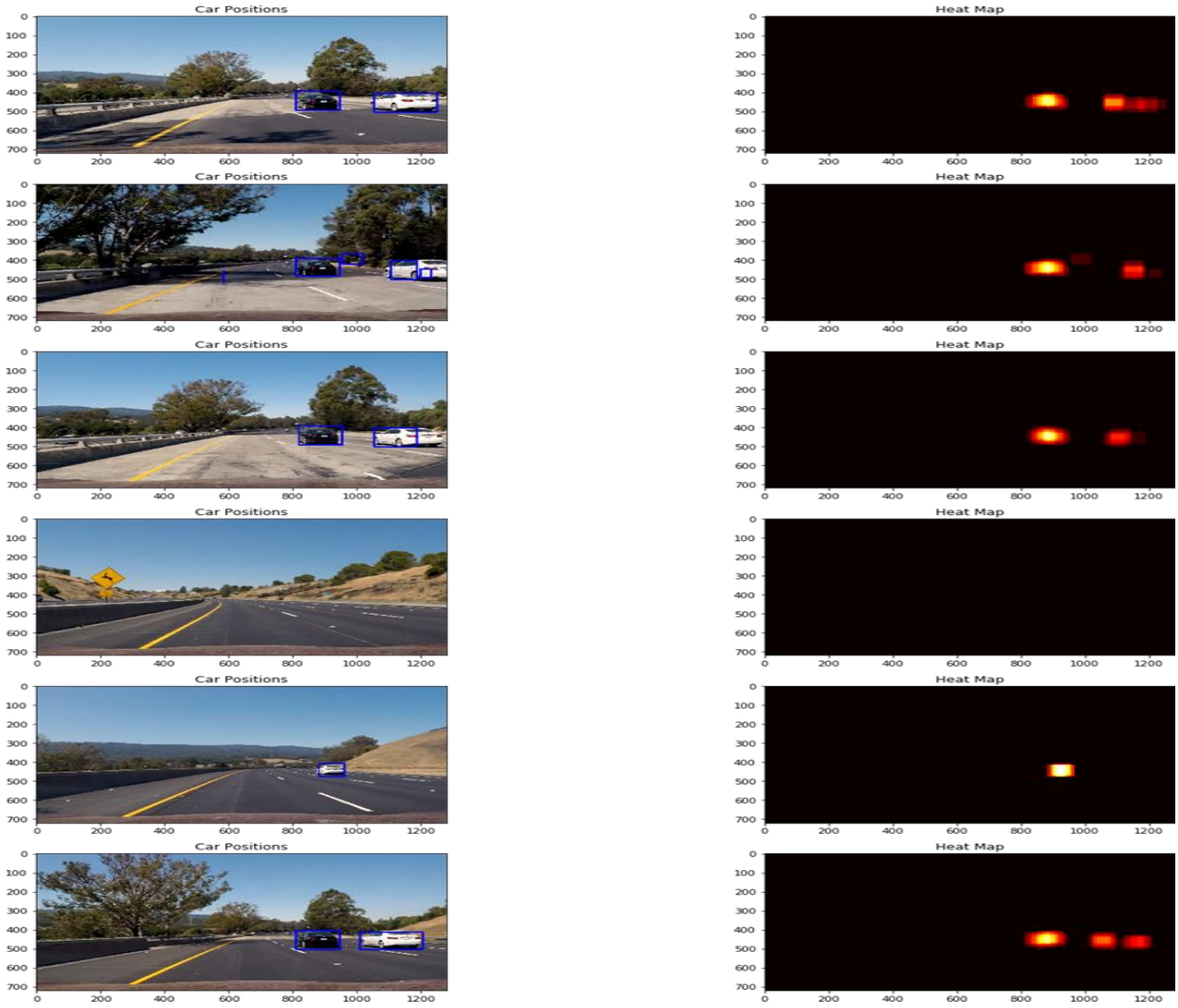

Fig. 6: Elimination of incorrect identifications and detection of numerous instances using heat maps.

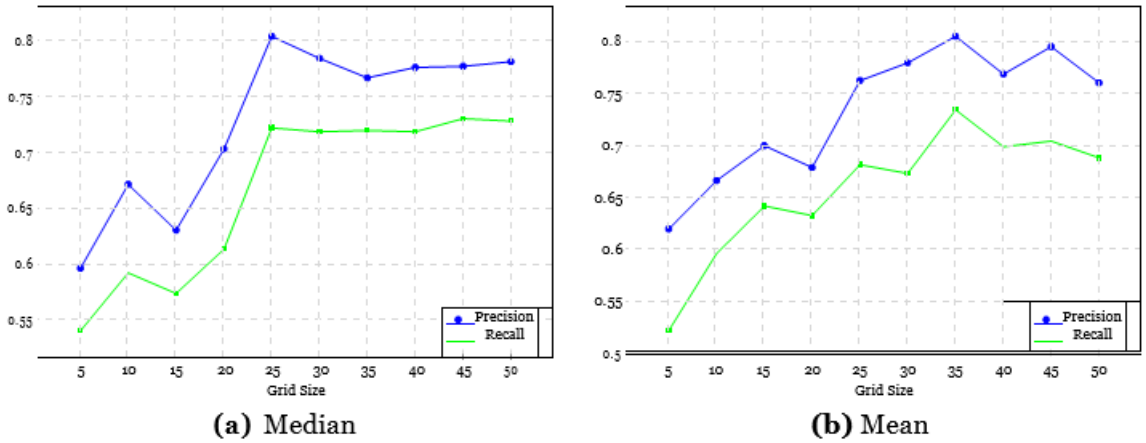

Fig. 7: Analysis of precision and recall over varying dimensions of rows and columns within the CIE L*a*b color space. (a) Outcomes obtained by utilizing the median values of the cells.(b) Outcomes obtained by utilizing the average cell values.

Fig. 8: The result of color quantization.

To obtain the median color value for each cell, it is optimal to divide the image into 25 columns and rows. According to Figure 9, the outcome is an accuracy of 80.43% and a recall of 72.22%. The inclusion of additional rows and columns hampers the efficiency of the procedure. The outcomes of computing the characteristics using the average instead of the middle value are depicted in Figure 10. Based on the suggested column and row divisions, the picture may be appropriately recognized with a recall rate of 73.47% and an accuracy rate of 80.55%. There is a 0.12% increase in accuracy and a 1.25% increase in recall compared to the median values. To get a recall of 73.79% and an accuracy of 80.91%, the optimal approach is to utilize 35 rows and columns in order to acquire the median color values. Compared to the ideal outcome achieved by utilizing the median in the CIE L*a*b color space, the recall exhibits a 1.57% rise, while the accuracy sees a 0.48% improvement. By utilizing the mean RGB values, the ideal configuration of 40 columns and rows enhances accuracy to 78.87% and recall to 72.70%. Because of the extensive number of dimensions in the required feature vector, this outcome is the most unfavorable as shown in Figure 9 and Figure 10.

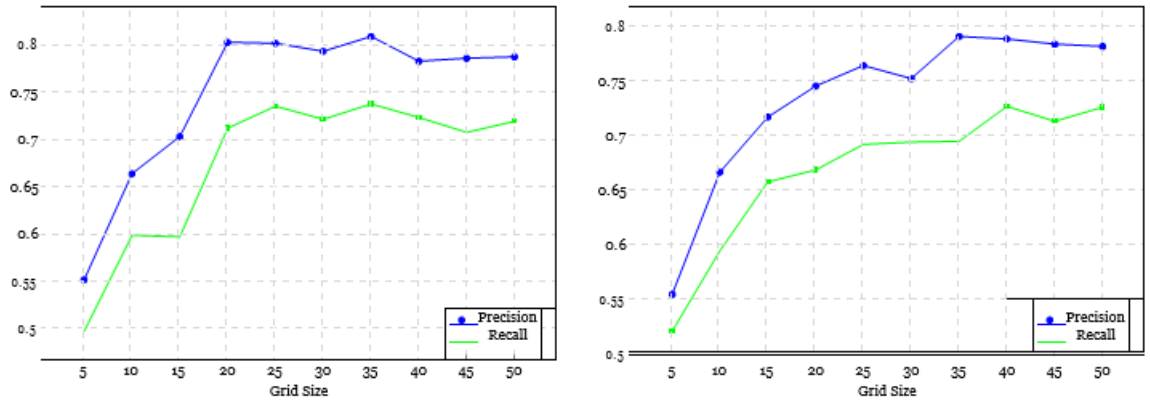

Fig. 9: Assessing the precision and recall metrics across varying dimensions of rows and columns within the RGB color space. (a) Outcomes were obtained by utilizing the median values of the cells. (b) Outcomes were obtained by utilizing the average cell values.

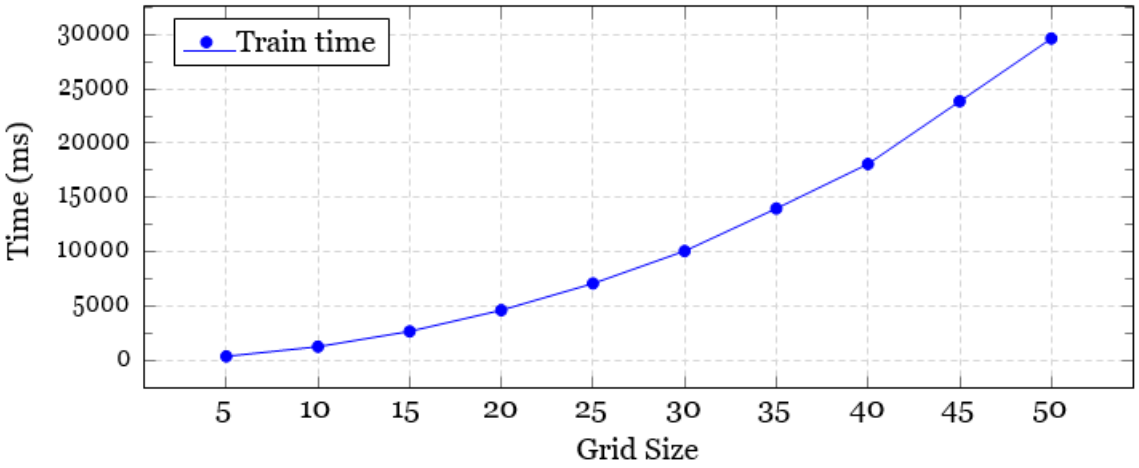

Fig. 10: Training durations for various grid sizes using median color values in the CIE L*a*b color space as features. The results shown are the average obtained from five iterations.

Table 1: Comparison work with literatures

Reference	Model	Accuracy
[12]	CNN	98.2%
[13]	CNN	99.5%
[14]	CNN	96%
Proposed work	CNN	99.7%

4. CONCLUSION

This paper, Using Single-Shot object Detection (SSD) and YOLO as benchmarks, assesses the effectiveness of two-space-free CNNs in terms of vehicle localization. The superior design structure of SSD enables it to outperform YOLO in all aspects, including evaluation and accuracy, assembly speed, and speculative capacity. SSD demonstrates superior performance compared to the initial model trained with 20 classes when trained with a reduced number of classes. Conversely, YOLO performs exceptionally poorly when just one class object is created. Further investigation of SSD models reveals that the organization becomes more proficient in achieving higher accuracy rates when it has several competing classes. Despite our lack of prior experience in machine learning, have made significant progress in creating an application capable of license plate identification and an OCR service for reading them. Due to our lack of prior understanding of neural networks, dedicated the initial weeks of our partnership to learning and familiarizing ourselves with this

subject. Both the theoretical and practical aspects of neural networks were taught during the class. During the study process, experimented with many methodologies and technologies, gaining insights from our errors, until identified the frameworks that would be most advantageous for us. During the testing phase, the completed solution accurately detected and recognized 99.7% of the license plates that met the specified quality standards. The percentage decreased to 6.95% when did not utilize our image preparation. I employed the TensorFlow and Keras frameworks within the Anaconda environment for Python to incorporate vehicle detection algorithms, picture registration, and YOLO object identification and recognition. This was achieved by utilizing the OpenCV library and employing deep learning methodologies.

. 4.1 Future Work

The process of designing and training neural networks is naturally characterized by iteration. Several obstacles might hinder the learning process, such as an insufficient optimization strategy, incomplete or mislabeled data, or insufficient data volume. Hence, given ample time and data, it is nearly always feasible to improve your model. Similarly, too might enhance our object detection and optical character recognition service if were given additional time. However, consider this to be superfluous given the diminishing yields it would generate. There are several potential trajectories for the future of this project. have established the foundation for developing a comparable service utilizing the same resources, notwithstanding the system's initial purpose for license plates. To achieve object detection for different objects, one can obtain photos of the desired item, train a Convolutional Neural Network (CNN) using the specified command, then adjust the model file within the software. The image preprocessing is specifically tailored to the shape and colors of license plates, rather than improving the readability of general objects. This makes it challenging to modify the OCR service for a different item. However, it is not flawless and there are techniques to enhance it. Due to time constraints, were unable to set up the OCR service on a server and establish a connection between our application and our employer's platform. Both prerequisites must be fulfilled prior to the project's utilization in future production.

References

1. K. Yamaguchi, Y. Nagaya, K. Ueda, H. Nemoto, and M. Nakagawa, ‘‘A method for identifying specific vehicles using template matching,’’ in Proc. IEEE Int. Conf. Intell. Transp. Syst., Oct. 2016, pp. 8–13.
2. B. B. Yousif, M. M. Ata, N. Fawzy, and M. Obaya, ‘‘Toward an optimized neutrosophic K-means with genetic algorithm for automatic vehicle license plate recognition (ONKM-AVLPR),’’ IEEE Access, vol. 8, pp. 49285–49312, 2020.
3. Abubaker, B. A., Ahmed, S. R., Guron, A. T., Fadhil, M., Algburi, S., & Abdulrahman, B. F. (2023, November). Spiking Neural Network for Enhanced Mobile Robots’ Navigation Control. In 2023 7th International Symposium on Innovative Approaches in Smart Technologies (ISAS) (pp. 1-8). IEEE..
4. C.-H. Lin, Y.-S. Lin, and W.-C. Liu, ‘‘An efficient license plate recognition system using convolution neural networks,’’ in Proc. IEEE Int. Conf. Appl. Syst. Invention (ICASI), Apr. 2018, pp. 224–227.
5. Waleed, M., Abdullah, A. S., & Ahmed, S. R. (2020, September). Classification of Vegetative pests for cucumber plants using artificial neural networks. In 2020 3rd International Conference on Engineering Technology and its Applications (IICETA) (pp. 47-51). IEEE.
6. S. S. Omran and J. A. Jarallah, ‘‘Iraqi car license plate recognition using OCR,’’ in Proc. Annu. Conf. New Trends Inf. Commun. Technol. Appl. (NTICT), Mar. 2017, pp. 298–303.

7. Mahmood, M. T., Ahmed, S. R. A., & Ahmed, M. R. A. (2020, November). Detection of vehicle with Infrared images in Road Traffic using YOLO computational mechanism. In IOP Conference Series: Materials Science and Engineering (Vol. 928, No. 2, p. 022027). IOP Publishing.
8. N. Rana and P. K. Dahiya, ‘‘Localization techniques in ANPR systems: A-state-of-art,’’ Int. J. Adv. Res. Comput. Sci. Softw. Eng., vol. 7, no. 5, pp. 682–686, May 2017.
9. Niu, Y., Habeeb, F. A., Mansoor, M. S. G., Gheni, H. M., Ahmed, S. R., & Radhi, A. D. (2022, October). A Photovoltaic Electric Vehicle Automatic Charging and Monitoring System. In 2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT) (pp. 241-246). IEEE..
10. Abdulateef, O. G., Abdullah, A. I., Ahmed, S. R., & Mahdi, M. S. (2022, October). Vehicle License Plate Detection Using Deep Learning. In 2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT) (pp. 288-292). IEEE..
11. D. Kim, T. Song, Y. Lee, and H. Ko, ‘‘Effective character segmentation for license plate recognition under illumination changing environment,’’ in Proc. IEEE Int. Conf. Consum. Electron. (ICCE), Jan. 2016, pp. 532–533.
12. Alam, N. A., Ahsan, M., Based, M. A., & Haider, J. (2021). Intelligent system for vehicles number plate detection and recognition using convolutional neural networks. Technologies, 9(1), 9.
13. N. Saif et al., "Automatic License Plate Recognition System for Bangla License Plates using Convolutional Neural Network," TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), Kochi, India, 2019, pp. 925-930, doi: 10.1109/TENCON.2019.8929280.
14. I. V. Pustokhina et al., "Automatic Vehicle License Plate Recognition Using Optimal K-Means With Convolutional Neural Network for Intelligent Transportation Systems," in IEEE Access, vol. 8, pp. 92907-92917, 2020