ANPD

AUTOMATIC NUMBER PLATE DETECTION

ABSTRACT

The rapid development and advancements in deep learning techniques have sparked significant interest among researchers worldwide, leading to the emergence of innovative and highly efficient number plate recognition systems. These systems, powered by state-of-the-art deep learning algorithms, have found extensive applications across various domains, including law enforcement, parking lot management, toll terminals, and traffic regulation. However, the existing implementations of these systems heavily rely on high-end computing resources, which pose challenges when deploying them on resource-constrained devices, particularly in the context of Internet of Things (IoT) devices characterized by limited memory and processing power.

One of the primary challenges faced by number plate recognition systems is the acquisition and processing of real-time Closed-Circuit Television (CCTV) footage, which is a highly time-consuming task. To overcome this challenge, the proposed work introduces an efficient deep learning model known as You Only Look Once (YOLO), which excels in object detection tasks. By leveraging the capabilities of YOLO, the proposed system aims to significantly reduce the computational requirements and processing time, making it suitable for real-time applications.

Automatic Number Plate detection (ANPD) serves as the central focus of the proposed work. ANPD is a widely adopted computer vision application that involves the extraction of number plate information from images or video frames. The proposed system aims to enhance the accuracy and efficiency of ANPD by utilizing deep learning techniques, specifically YOLO, for robust and reliable object detection.

Furthermore, the applications of number plate recognition systems are multifaceted and have far-reaching implications. In the realm of law enforcement, these systems play a pivotal role in identifying and tracking vehicles involved in criminal activities, thereby aiding in crime prevention and detection. Parking lot management systems benefit from number plate recognition by automating entry and exit processes, monitoring parking durations, and enforcing parking regulations effectively.

By addressing the challenges posed by resource limitations and the time-consuming nature of real-time CCTV footage acquisition, the proposed work seeks to pave the way for more efficient and practical implementation of number plate recognition systems. The integration of YOLO into ANPD holds immense promise for enhancing the accuracy, speed, and effectiveness of number plate recognition, thereby enabling its widespread adoption across various industries and domains.

TABLE OF CONTENTS

I. INTRODUCTION

Deep learning and neural networks have gained momentum in the past few years, which has led to the development of automated licence plate Detection (ANPD). This can be used in public places to monitor things like traffic safety enforcement, automatic toll tax collection, car park systems, and automatic vehicle parking systems. Traffic security and congestion represent some of the major problems in upcoming smart cities. Any contribution to help manage these cities will be beneficial for everyone.

With the further development of smart cities and consequently intelligent transportation systems, efficient automatic number plate recognition systems are now required more than ever. These systems have gained much attraction due to their application in intelligent surveillance systems which have various use cases such as automated parking lot management, traffic surveillance, vehicular access control, etc., which represent emerging research areas under the scope of urban mobility.

Object detection software often use a combination of natural language processing and deep learning approaches to accomplish this task. CNN is popular because of its versatility in jobs like picture categorization and object recognition. A neural network that is built on specific regions is known as a regional neural network. Due to its processing complexity, it is not appropriate for real-time application. complexity of the application YOLO (You Only Live Once) architecture is a popular term for "You Only Look Once" architecture. It's a more efficient architecture that can detect objects in real time.

The system of our proposed system, which is used for number plate detection, is based on the recently released YOLOv5which is a time-efficient successor of You Look Only Once advanced deep learning object detection architecture. For this system, we chose the lightweight version of Yolov5, named v5 small, which consists of 283 layers, 16.4 GFLOPS, and about 7 million parameters.

1.1 OBJECTIVE

The objective of implementing ALPD using YOLO is to create a robust and efficient system that can accurately detect and recognize license plates in real-time or near-real-time from images or video streams.

Utilize the YOLO object detection algorithm to accurately locate and extract license plates from input images or video frames. It is to achieve high precision and recall in license plate detection, even in challenging conditions such as low resolution, occlusion, or varying angles. Optimized the YOLO-based ALPD system for real-time or near-real-time processing. The objective is to achieve fast inference speeds that can handle high volumes of video frames or images in real-world scenarios.

Training the YOLO model on the prepared dataset and fine-tune it specifically for license plate detection and recognition tasks. It is to optimize the model's performance by adjusting hyperparameters, incorporating data augmentation techniques, and balancing the trade-off between speed and accuracy.

Integrate the trained YOLO-based ALPD model into a practical software or hardware system. The objective is to create a user-friendly system that can be easily deployed in various applications, such as surveillance, traffic management, or access control. Overall, The objective of ALPD is to create a system that increases the security in various fields by automated detections of the number plate.

1.2 MOTIVATION

The motivation behind ALPD (Automatic License Plate Detection) stems from the need for efficient and accurate identification and tracking of vehicles in various applications.

ALPD plays a crucial role in law enforcement by enabling the identification of vehicles involved in criminal activities or traffic violations. It assists in detecting stolen vehicles, tracking suspects, and enforcing traffic regulations. ALPD contributes to effective traffic management by providing real-time data on vehicle movements, traffic flow, and congestion. This information helps optimize traffic control, improve road safety, and facilitate intelligent transportation systems.

These systems are employed in parking facilities to automate the identification of vehicles entering or exiting parking areas. It assists in monitoring parking occupancy, enforcing parking regulations, and facilitating efficient parking space allocation. It is utilized in toll collection systems to automate the identification and billing of vehicles passing through toll booths. It enables seamless and efficient toll payment processes, reducing congestion and manual interventions.

ALPD enhances surveillance and security systems by enabling the identification and tracking of vehicles of interest. It helps in monitoring sensitive areas, identifying unauthorized vehicles, and supporting investigations. It is also utilized in access control systems to automate entry or exit processes in secured areas such as gated communities, corporate premises, or restricted zones. It ensures efficient and secure access management. Overall, the motivation behind ALPD is to leverage technology to streamline vehicle identification, improve operational efficiency, enhance safety, and enable data-driven decision-making in various domains related to transportation and security.

II. RELATED WORKS

Vehicle license plate detection and recognition have been compelling areas of research for a long time. Computer vision researchers have devised many different approaches to solve the task at hand.

Over the past few years, many researchers have addressed the license plate detection task. For example, the authors of **[1]** used ResNet-50-based Faster-RCNN for Indian license plate detection and achieved a very high mAP. Mean Average Precision (mAP) is a commonly used measurement of precision for object detection models and measures the crossing of the predicted bounding box with the labelled bounding box.

The authors of **[2]** proposed a Fast-YOLO **[3]** and YOLOv2 **[4]-**based detector. In this approach, the authors trained two CNNs, first for vehicle detection and another for license plate detection. Their results suggested that Fast-Yolo had impressive results in both tasks.

On the other hand, the authors of **[5]** used an exemplar SVM-based approach along with Fast and Faster-RCNN, depicting that RCNN was better suited for real-time detection. The authors of **[6]** also presented a YOLOv2 and Fast YOLO-based single cascaded CNN to detect both car frontal car views and licence plates, achieving high recall and precision rates. The authors of **[7]** used 500 images to train an AlexNet neural network and to run it on the Jetson TX1 board. Although they achieved a high accuracy, their testing set only consisted of 64 images.

III. DATASET DESCRIPTION

The dataset that we used for this paper is a combination of number plate data from the Google open Images dataset and the Indian number plates dataset that is available on Kaggle. The original dataset is about 200 images.

We manually added the labels for these images using Makesense.ai website. All the images in this dataset have a text label file embedded with them. This label file contains bounding box coordinates for a number plate in YOLO annotation format. YOLO accepts the labels in a certain format given below:

<object-id> <center_x> <center_y> <width> <height>

where:
object-id: number corresponding to object category;
Center_x and Center_y: normalized center point of the bounding box;
Width and height: normalized width and height of the bounding box.

After acquiring proper dataset, We have used Robo-flow website for pre-processing of our dataset. We have performed a few Augmentation Steps for better detection. The final training dataset is of 678MB of size and contains 400+ images. The augmentation steps include:

1. Rotations
2. Gray-scaling
3. Brightness
4. Noisy images

The dataset was divided into 3 folders, namely train, test, and validation. The training set contained 456 samples which roughly constituted 70% of the dataset. The validation set that was used to validate the model performance while training contained 250 images which composed nearly 20% of the database. Lastly, the test dataset which was used to perform the final testing of the model contained 160 images.

Fig-1: *Augmentation steps*

Coordinates of the example bounding box in YOLO format are:
[((420 + 98) / 2) / 640, ((462 + 345) / 2) / 480, 322 / 640, 117 / 480] which are
[x_center, y_center, width, height] : [0.4046875, 0.840625, 0.503125, 0.24375].

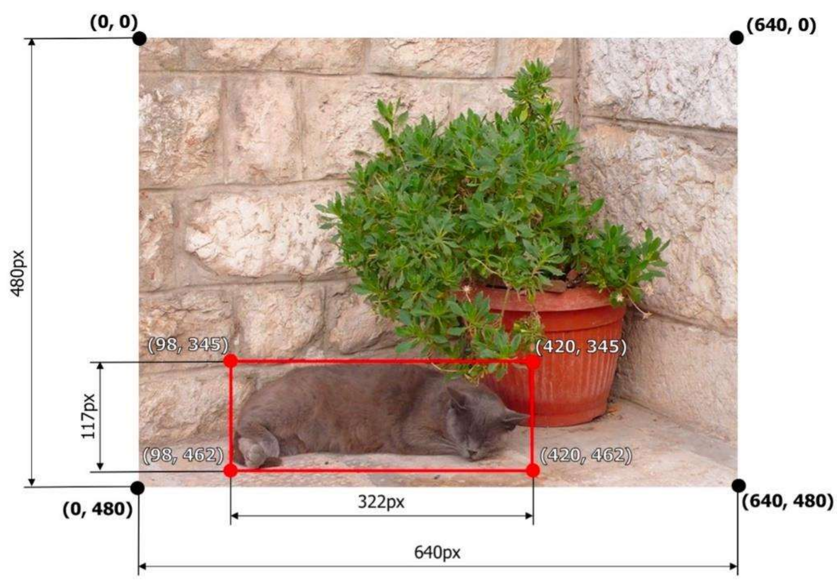

Fig-2: *Bonding Box format for YOLO*

IV. PROPOSED METHOD

The beginning of this work is to obtain the license of the vehicle from the photograph or video. Object detection usually has two steps:

1) object localization and

2) object classification.

Object localization refers to the presence of an object in an image or video, and object classification, whether it is a car, bus or truck, etc. It refers to the classification of objects.

In our project, we use the YOLOv5 model, which is the latest version of YOLO, a one-step deep learning object detection model. YOLO means "You Only See One". The model, as the name suggests, looks at the image only once. Among all existing algorithms, YOLO is cons idered the fastest model for detecting real objects. The model is GPUcentric, meaning it uses a single GPU.

4.1 Network Design:

Our model Architecture is that the input image features will be compressed down through feature extractor (Backbone) and then forwarding to object detector (including Detection Neck and Detection Head).

Detection Neck (or Neck) works as a feature aggregation which is tasked to mix and combine the features formed in the Backbone to prepare for the detection step in Detection Head (or Head).

1) **Model Backbone:**

   The backbone of a model is the element dedicated to taking the input image and extracting feature maps from it. This includes ensuring the right feature map dimensions, which at times requires slightly modifying the scaling factor for the width and depth of the model.

   The backbone obtains feature maps of different sizes, and then fuses these features through the feature fusion network (neck) to finally generate three feature maps P3, P4, and P5 to detect small, medium, and large objects in the picture.

   The CSP (Cross Stage Partial networks) are the main backbone for feature extraction. These are based on the Densenet, which is mainly used to connect CNN layers.

2) Model Neck:

The term ‘neck’, the structure placed between the head and backbone whose objective is to aggregate as much information extracted by the backbone as possible before it is fed to the head. Neck does features aggregation to make inter-relation between necessary features in order to get better result.

The image features are processed into semantical features. In other words, from the low-level layers, the deeper that the input image goes through, the complexity of semantical features will be more increased while the spatial resolution of feature maps will be more decreases due to down-sampling.

Feature pyramids are used to recognise elements in the image of varying sizes and produce multi-scale predictions. This is achieved through the usage of feature pyramid networks. YOLOv5 makes use of PaNets for FPN (Feature Pyramid Networks).

3) Model Head:

The function of the head is to perform predictions. The prediction is the final prediction composed of a vector containing the predicted bounding box coordinates (center, height, width).

The prediction confidence score, and the probability classes. YOLO deploys the identical head for detection with the anchor-based detection steps. Anchor box is a list of predefined boxes that best match the desired objects. The bounding boxes were not only predicted based on ground truth boxes but also predefined anchor boxes.

The final detection stage is Model Head, which leverages anchor to locate vectors with class probabilities, objectness scores, and bounding boxes. In a deep neural network, activation functions are very significant. The YOLOv5 model uses the Pytorch framework and includes 191 layers.

4.2 Reasons for Choosing YOLOv5:

The following are the major reasons for using the YOLOv5 detection model in our project: With YOLOv5, we can have custom object detection by training the custom datasets. It gives us different models to choose from according to our custom datasets. It also uses data augmentation, which combines images to give new data, which can help the model perform well with new data. It combines K-means with a genetic algorithm to give the K-means evolved anchor boxes.

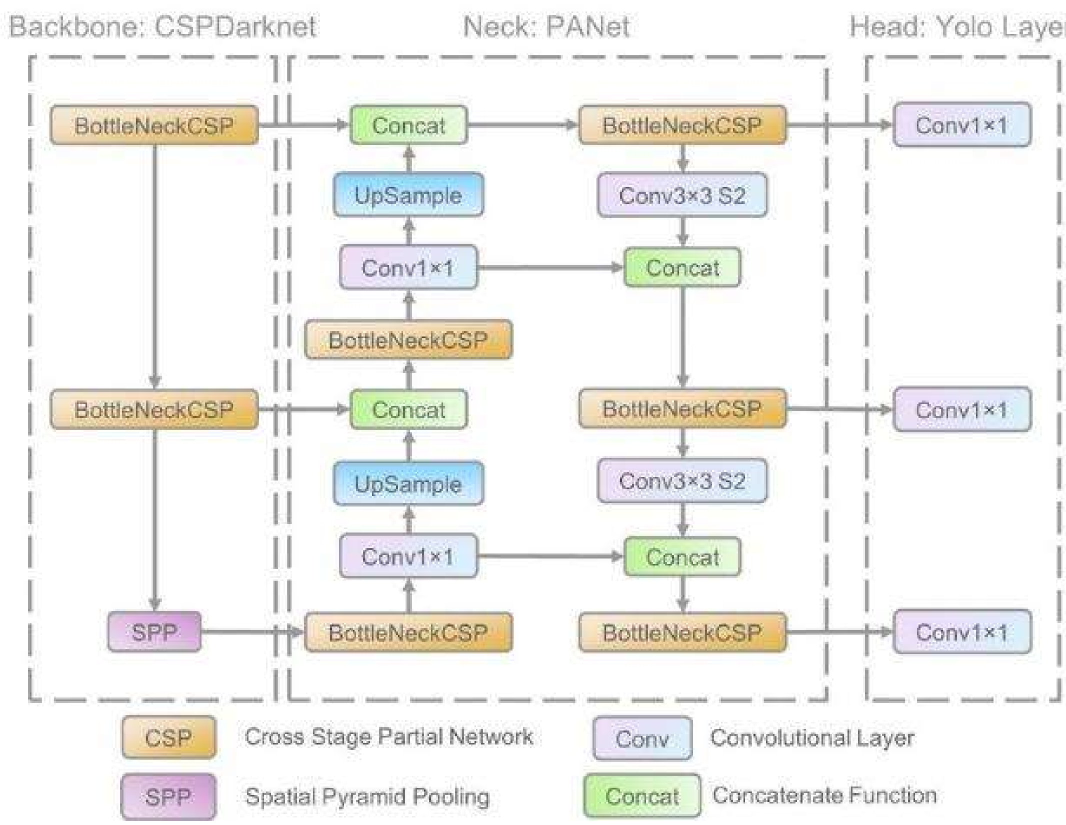

Fig-3: *Network Architecture of YOLOv5*

V. EXPERIMENT RESULT AND ANALYSIS

5.1 IMPLEMENTATION:

Object Detection pipeline consists of 3 parts: Training, Validation and Testing

Training:

YOLOv5s was trained with following hyperparameters:

- Input Image Size: 640
- Batch Size: 16
- Epochs: 300 (converged in 198 epochs)
- Pretrained Weights: yolov5s.pt

The training dataset consisted of 400 images along with their class and bounding box details mentioned in a txt file in yolo format. The training was set to run for 300 epochs but the model converged in 198 epochs and the training was stopped.

Validation:

The validation dataset consisted of 100 images along with their class and bounding box details in a txt file in yolo format for validation purpose. In validation phase the model reached Mean Average Precision (mAP) of 0.91.

Following are some of the image results generated in validation phase:

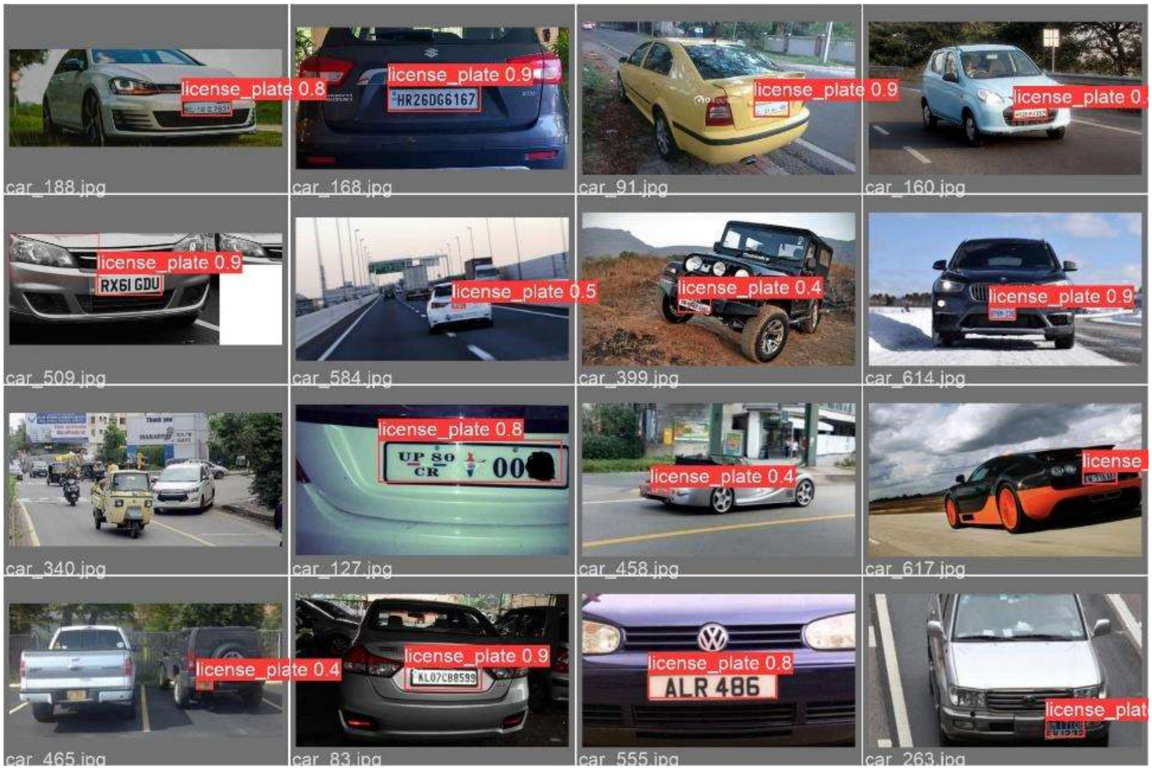

Fig-4: *Image results in validation phase*

At the end of training and validation epochs, a weights file ("best.pt") is generated which consists of all the learned parameters of the model.

Testing Phase:

The model was tested on various images and videos and the model generated accurate class and bounding box predictions. The weights file called "best.pt" that was generate in the training phase was used for inference in testing phase. Testing was carried out in PyTorch.

5.2 PERFORMENCE METRICS:

For object detection tasks, mAP or mean average precision is often the performance metric of choice for many computer vision practitioners. Object detection tasks include identifying the relevant object in the images and then stratifying it into known classes. These tasks make predictions in the form of object class labels and a bounding box surrounding the object itself. Precision and recall are the two-core metrics that together were used to evaluate the mAP or performance of the model.

Precision is a measure of quality. It describes how often our model guessed correctly from all the guesses. Perfect precision means every guess is correct. On the other hand, recall is a measure of quantity; it indicates whether our model predicted the same result as expected or not. Perfect recall indicates that a model has made the correct number of predictions or has given full coverage.

Precision = TP/(TP + FP)
Recall = TP/(TP + FN)

where:
TP: True positives;
TN: True negatives;
FP: False positives;
FN: False negatives.

The mAP was calculated by computing a series of precision-recall curves (AP) over varying Intersection over Union (IoU) levels. IoU is another performance metric that measures the accuracy of the bounding box on a particular task. In other words, IoU measures the overlap between 2 annotations. First, the average precision per class is calculated, and then it is averaged over all object categories to obtain mAP.

5.3 RESULTS:

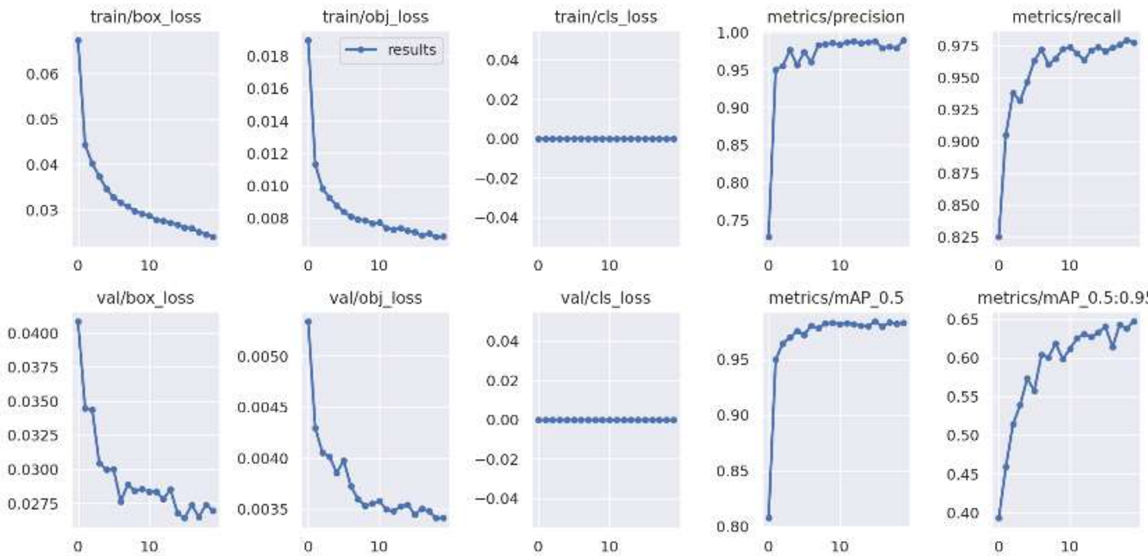

Fig-5: *Precision and Loss of Yolov5 model*

Fig-6: *Vehicle number plate detection*

VI. ABBREVATIONS AND REFERNCES

6.1 Abbreviations

ANPD	Automatic number plate detection
YOLO	You only look once
IoT	Internet of things
mAP	Mean Average precision
CNN	Convolutional Neural network
CCTV	Closed-Circuit Television
RCNN	Region based CNN
GPU	Graphics processing Unit
CSP	Cross stage partial
FPN	Feature pyramid network
IoU	Intersection over union

6.2 References

1. Ravirathinam, P.; Patawari, A. Automatic License Plate Recognition for Indian Roads Using Faster-RCNN. In Proceedings of the 11th International Conference on Advanced Computing (ICoAC), Chennai, India, 18–20 December 2019; pp. 275–281.

2. Laroca, R.; Severo, E.; Zanlorensi, L.A.; Oliveira, L.S.; Goncalves, G.R.; Schwartz, W.R.; Menotti, D. A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector. In Proceedings of the 2018 International Joint Conference on Neural Networks (Ijcnn), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–10.

3. Shaifee, M.J.; Chywl, B.; Li, F.; Wong, A. Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video. *arXiv* **2017**, arXiv:1709.05943.]

4. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21 July 2017; pp. 7263–7271.

5. Rafique, M.A.; Pedrycz, W.; Jeon, M. Vehicle license plate detection using region-based convolutional neural networks. *Soft Comput.* **2018**, *22*, 6429–6440.

6. Montazzolli, S.; Jung, C. Real-time brazilian license plate detection and recognition using deep con-volutional neural networks. In Proceedings of the 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), Niterói, Brazil, 17 October 2017; pp. 55–62.

7. Lee, S.; Son, K.; Kim, H.; Park, J. Car plate recognition based on CNN using embedded system with GPU. In Proceedings of the 10th International Conference on Human System Interactions (HSI), Ulsan, Korea, 17–19 June 2017; pp. 239–241.