T.C.

ALTINBAS UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

VEHICLE POSITION ESTIMATION AND VEHICLE CLASSIFICATION USING DEEP CONVOLUTIONAL NEURAL NETWORKS

Bashaer Isam Hasan KABEAYLA

Master Thesis

Supervisor

Assoc. Prof. Dr. Yasa EKŞİOĞLU ÖZOK

Istanbul, 2021

VEHICLE POSITION ESTIMATION AND VEHICLE CLASSIFICATION USING DEEP CONVOLUTIONAL NEURAL NETWORKS

by

Bashaer Isam Hasan KABEAYLA

Electrical and Computer Engineering

Submitted to the Institute of Graduate Studies

in partial fulfillment of the requirements for the degree of

Master of Science

ALTINBAŞ UNIVERSITY

2021

The thesis titled **"VEHICLE POSITION ESTIMATION AND VEHICLE CLASSIFICATION USING DEEP CONVOLUTIONAL NEURAL NETWORKS"** prepared and presented by Bashaer Isam Hasan KABEAYLA was accepted as a Master of Science Thesis in Electrical and Computer Engineering.

Assoc. Prof. Dr. Yasa EKŞİOĞLU ÖZOK

Supervisor

Thesis Defense Jury Members:

Assoc. Prof. Dr. Yasa EKŞİOĞLU ÖZOK	School of Engineering and Natural Sciences, Altinbas University	_____
Asst. Prof. Dr. Doğu Çağdaş ATİLLA	School of Engineering and Natural Sciences, Altinbas University	_____
Assoc. Prof. Dr. Aydın Tarık ZENGİN	Faculty of Sports Sciences, Istanbul Zaim University	_____

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Approval Date of Institute of Graduate Studies: ____/____/____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

BASHAER ISAM HASAN KABEAYLA

DEDICATION

I devote and pledge this research work to my supervisor who is salient for guiding me through whole research work as well as my family for always assisting me in my hard time.

ACKNOWLEDGEMENTS

First and foremost I would like to thank my supervisor for guiding and helping me along the way in writing this dissertation. Discussing my progress, problems, and ideas with my supervisor a couple of times every week helped me tremendously in understanding the rationale behind the assignment. It made me better realize the business need for this analysis. He helped me by looking outside of merely all the software project management information, and considering the organization from a more holistic perspective than just from the productivity point of view. He patiently explained to me a great number of technical details with regards to neural networks, for example how to deal with such an unstable learning method and to provide insight into the reliability of those results. Without their time and help none of this research would have been possible, and I'm grateful to them for giving me the opportunity to work on this interesting subject.

ABSTRACT

VEHICLE POSITION ESTIMATION AND VEHICLE CLASSIFICATION USING DEEP CONVOLUTIONAL NEURAL NETWORKS

KABEAYLA, Bashaer Isam Hasan,

M.Sc., Electrical and Computer Engineering, Altınbaş University,

Supervisor: Assoc. Prof. Dr. Yasa EKŞİOĞLU ÖZOK

Date: July/2021

Pages: 64

The aim of this master's thesis is to classify the vehicles and estimate the position with license plate localization using Deep Convolutional Neural Network (DCNN). Vehicle pose estimation with license plate localization serves as one of the most widely-used real-world applications in fields like toll control, traffic scene analysis, and suspected vehicle tracking. Along with license plate information, to obtain overall comprehension, the information of the owner vehicle also plays a great role, and contextual information is defined as the relationship between the vehicles pose license plate and the owner vehicle in our work. We proposed a one-stage anchor-free vehicle classifier for simultaneously localizing the region of license plates and vehicles' poses. The classifier, rather than bounding rectangles, gives bounding quadrilaterals, which gives a more precise indication for vehicle pose estimation with license plates localization. For single scale input, we reached mean Precision Accuracy mAP/mAP50 of 35.4/82.3 on the Laboratory for Intelligence and Safe Automobile (LISA) benchmark dataset, already outperformed the existing commercial systems OpenALPR and Sighthound. For multi-scale input, we reached the best mAP/mAP50 of 40.8/90.1. For the vehicle pose (front-rear), classification accuracy reached

98.8%, average IoU reached 71.3%, giving a promising result as an end-to-end vehicle position estimation and license plate localization with contextual information. The work has performed in python programming language with several libraries of deep learning were being used for this purpose. There are three functional head networks in our design, and thus the end-to-end and simultaneous training leads to a potential training instability. If the model fails to converge, it is quite hard to trace which head design is raising a problem. Thus, in our design process, we added each the functional head step-by-step, making sure the single head design idea is working correctly and then expanded the model with the rest functional heads. Our DCNN model training started from an initial weight which we had already trained for about 110000 iterations in the model without classification head, so the total training iterations will be around 780000 including the transfer learning part in DCNN. Transfer learning made the DCNN model start at a smart point and made it easier to optimize all of the functional heads simultaneously.

Keywords: Vehicle classification, pose estimation, optimization, DCNN, license plate.

TABLE OF CONTENTS

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

LIST OF ABBREVIATIONS

AUC	:	Area Under Consideration
CAD	:	Computer Aided Design
CNN	:	Convolutional Neural Network
DCNN	:	Deep Convolutional Neural Network
FR		Front Rear
GT	:	Ground Truth
LISA	:	Laboratory for Intelligence and Safe Automobile
mAP	:	Mean Average Precision
OCR	:	Optical Character Recognition
OHEM	:	Online Hard Example Mining
ROI	:	Region of Interest

1. INTRODUCTION

The broader goal of this thesis is to investigate how neural systems can stay robust to incomplete information for vehicle classification and position estimation with license plate localization. Humans are naturally gifted at extracting knowledge and taking decisions based on imperfect observations about the state of their environment. The evolutionary advantage of developing this ability is hardly questionable, however it remains to be clearly understood how exactly our brains can achieve this. Besides of being of great interest to improve our understanding of learning and information processing in neurobiology, this question will awaken interest in any engineer attempting to design more robust intelligent systems. In the context of this thesis, the focus will be kept on the second perspective by concentrating on the study of artificial neural networks. The specific problem of interest will be the recognition of vehicles in occluded images. This is a strategic choice for research on this topic, because classical vehicle recognition is a well-researched territory, both in the brain as well as in artificial systems. This is why it will be useful to review the current state of the field and introduce important theoretical notions before expanding on the core of the thesis. Vehicle detection has a noticeable evolution after deep learning appeared and aided vehicle detection. After 2016, the Pascal Visual Object Class (VOC) in vehicle detection challenge has been dominated by deep learning-based methods. Followed by this trend, many application fields related to vehicle detection are also benefited. Robot vision, monitoring system (e.g., face detection), self-driving car (e.g., pedestrian detection), and our main research topic, license plate detection, all catch up with the trend and are all further improving due to the aid of deep learning. The visual system belongs to the most frequently researched and best understood parts of the brain. A natural preference for scientists to investigate the mechanisms of vision could be justified by the predominance of visual input among all sensory senses in humans. Another rationale for the appeal of studying the visual system is that vision is one of the easiest senses to manipulate in a controlled experiment.

Today, vehicle classification with deep convolutional neural networks is very important task. This all started with the design of a neural network model by researchers in the 1980s [1]. This network model nicknamed the “neocognitron” was an attempt by researchers to emulate the visual information processing of the vehicle position stream. For thisresearchers sequentially stacked, what are nowadays called convolutional neural network layers. This model was especially elegant, because it was efficient and respecting the biological models of

the time. First of all, it had hierarchical feedforward architecture. Secondly, because convolutional layers implement a very local connectivity pattern between layers, it meant that neurons in the upper layers would have receptive fields of increasing sizes compared to neurons in lower layers. In addition, the use of convolutional layers solved the problem of translation invariance, not only biologically relevant [2], but also generally desired for a vehicle detection network. Another benefit of the convolutional architecture was that neurons were not connected to all the other neurons from neighboring layers, as is the case in densely connected networks. This downsized connectivity and the weight sharing characteristic of convolutional networks produce a reduction in the size of the weight search space, which simplifies the network optimization procedure. Although training algorithms of the time were incompatible with deep multilayer networks, the neocognitron could achieve good performances for very simple digit recognition tasks [3] by combining an unsupervised learning algorithm for the internal layers [4] and basic perceptron supervised learning for the last layer [5]. Once this type of convolutional architectures was combined with supervised learning techniques based on backpropagation [6], the field of computer vision could start to get serious about vehicle recognition. In the 1990s researchers began to successfully train systems that could recognize handwritten numbers [7] and dataset of images depicting different classes of vehicles [8].

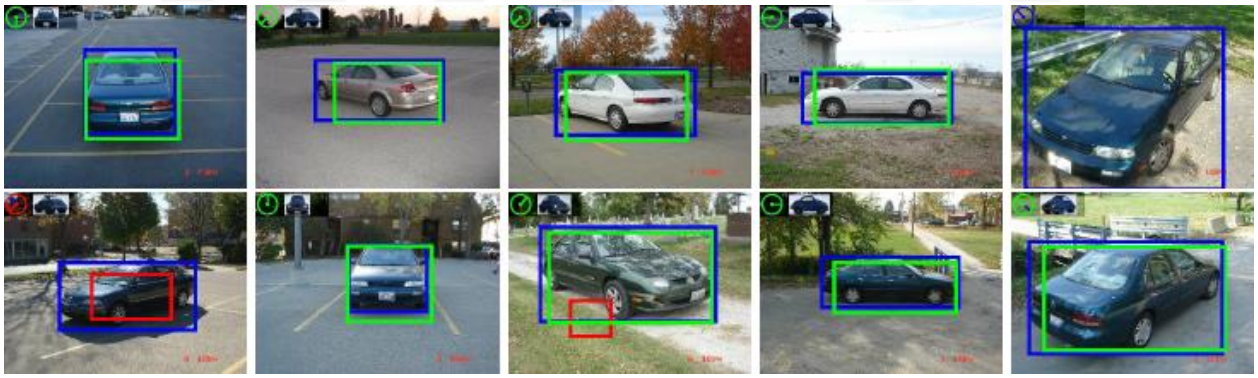


Figure 1.1: Pose-estimation previously performed on the UIUC dataset [9].

1.1 PROBLEM STATEMENT

In the vehicle pose estimation stage, most existing systems are based on the frontal view of vehicle. The open-source version of commercial software struggles to find the vehicle pose estimation and license plate with transformation, like a tilted license plate or the license plate which belongs to the car in an oblique view. This problem states a critical performance

bottleneck for machine learning based systems since the vehicle position estimation and detection failure means the total inability to comprehend a specific position.

The method proposed by author in [10] aimed to detect the vehicle position in a more general way, their model first detects cars inside an image and further finds the position with license plates for those cars, this improved the ability to find tilted vehicle classification with plates in pictures, as the high accuracy car detection has been developed recently. Their method highly reduced the detection missing (false-negative cases) of the license plate and position of vehicle, but the drawback exists, within the pictures with multiple cars inside, their model might regress to a license plate which doesn't belong to the vehicle and cause a loss of contextual information between the vehicle and the position estimation, Here, we conclude two problems of vehicle pose estimation we would like to solve:

1. Lack of estimation ability for tilted and oblique image of vehicles in real time.
2. Loss of contextual information while dealing with more general vehicle classification and position estimation.
3. Is it possible to generate training data by using a vehicle image and a measured ground truth of average vehicle classification?
4. Does the approach of deep learning techniques as a way of automating vehicle classification and position estimation measurements with the help of real time images?
5. What aspects need to be considered if further work on the subject is to be performed?

1.2 AID OF DEEP LEARNING

Since the classic deep learning pipeline is divided into four sub-tasks, the optimization process is quite tricky as each of the processes links to and counts on each other. An overall and end-to-end optimization strategy is needed if we want to push the performance of deep learning to another level.

Recently, deep learning technics aid the deep learning a lot especially with the help of Convolutional Neural Network (CNN). Researchers in [11] proposed a network named Warped Planar Vehicle Detection Network, their method is to find a license plate inside a single vehicle using Deep Convolutional Neural Network (DCNN), with obtaining not only rectangle bounding boxes, but also parallelograms after the model learned the affine transform

parameters and applied to rectangle bounding boxes. Their model can transform the license plate back into the rectangle easily by inverse affine transform and further do the DCNN.

Researchers in [12] designed a single network handling both license plate segmentation and Optical Character Recognition (OCR) at the same time and leads to a possibility of end-to-end optimization for segmentation and character recognition.

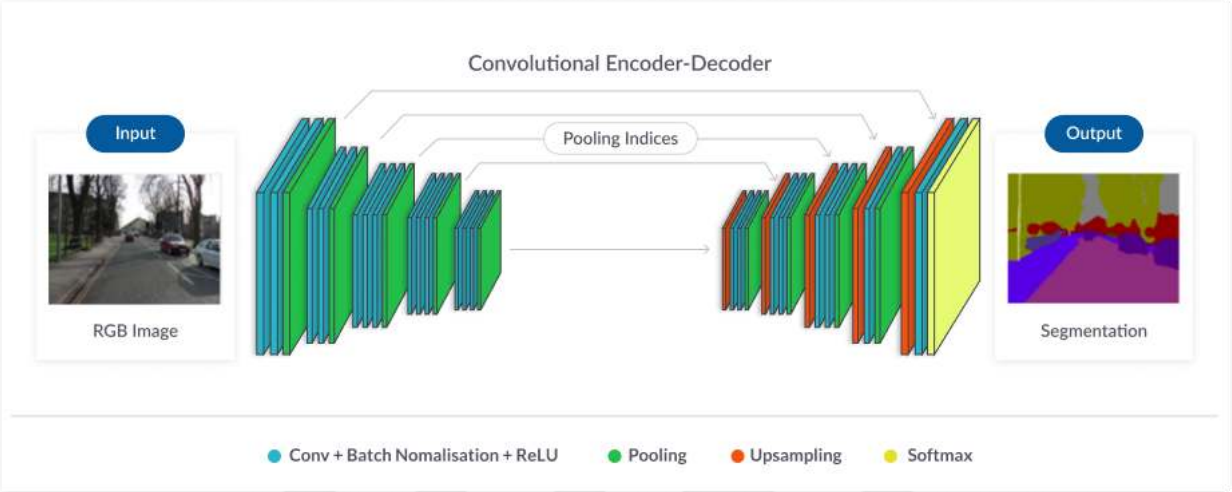


Figure 1.2: Image segmentation using deep learning [12].

1.3 CONTRIBUTION

There are four sub-tasks in the vehicle classification and vehicle pose estimation with license plate localization process, our research focuses on the vehicle pose estimation with license plate localization after vehicle classification task, aiming to accurately locate the vehicle position in various scenes with single or multiple vehicles inside. As to maintain the contextual information when dealing with higher vehicle classification performance, the license plate localization will be detected as well, the pose of the car, on which side (front or rear) does the license plate lay on will also be classified. We will show a detection example; the vehicle poses with license plate localization missing problem mentioned in the problem statement can be solved by our proposed method.

We conclude our main contributions as followed:

- We designed a DCNN-based network for vehicle classification and pose estimation vertices with a variety of angles in real-world scenes.

- Along with vehicle pose estimation and the license plate localization, vehicle's pose information will also be given.
- Our model will detect and classify the region of the car's front part and the rear part where a vehicle position is onto.
- To train our model, we proposed a new dataset with manually annotated front-rear bounding boxes.
- The classification process is based on a novel anchor-free method presented in methodology, no manually decided anchor-box size is needed; it might be an inspiration for other vehicle classification and detecting applications.

1.4 STRUCTURE OF THESIS

Followed by Chapter 1, Chapter 2 will introduce the background knowledge related to our work, including the recent vehicle classifier proposed in literature until 2019, the backbone architecture designed for vehicle classification and position estimation with license plate localization, a loss function for adjusting the weight between difficult learning case and easy learning case , and a post-processing technic often used for eliminating duplicate detection results

Chapter 3 gives a full understanding of our proposed method, including the whole model design, the optimization strategy with loss function design and detail training label encoding implementation, and training tricks with explanation of the reasons why we applied them in the training process.

Chapter 4 presents the performance of our model, with the dataset used for training and validation, quantitative mean Average Precision (mAP) analysis and qualitative results, followed by the discussion of how the detection ability of our model differed during the training process. Chapter 5 provides the discussion and in Chapter 6 we conclude our work.

2. RELATED WORK

This chapter reviews the technics related to deep learning vehicle classification, the designing pipeline of a vehicle position estimation and classifier includes:

1. Purpose-oriented model architecture design.

2. Training methods, which involve a broad range of topics, for example, the choice of quality training data and handling the inevitable over-fitting problems in deep learning studies. For vehicle detection, a procedure called hard negative mining is especially crucial since unlike classification problem, the training process of a vehicle classifier will have a bunch of background samples (or called negative samples) and relative-small amount of positive samples, hard negative mining is the method for dealing with the unbalanced sample amount between positive and negative ones which may hurt the classifier's performance.

3. Post-processing procedure for eliminating the duplicated detection results.

2.1 VEHICLE CLASSIFIER

A popular benchmark for vehicle detection is the Pascal Visual Vehicle Class (VVC) challenge, the version released in 2012 [13] includes 11,530 images with 20 classes and 27,450 bounding box annotations in the training and validation dataset. The Figure 2.1 shows the mAP increasing trend for the Pascal VOC challenge, the first three methods are based on traditional visual descriptors like Histogram of Gradient (HOG) used in Support Vector Machine - Histogram of Oriented Gradient (SVM-HOG) and Scale-Invariant Feature Transform (SIFT) used in Deformable Part Model - Math Kernel Library (DPM-MKL) in python.

After 2013, deep learning-based detection methods almost dominate the Pascal VOC challenge, including Region-based Convolutional Neural Network (R-CNN) [14] and its extensions Fast R-CNN [15] and Faster R-CNN [16], Single Shot Detector (SSD) [17] and its extension De-convolutional Single Shot Detector (DSSD) [17], after 2017, the newly released state-of-the-art classifier are still improving their detecting ability but tend to shift to another more challenging benchmark Microsoft - Common Objects in Context (MS-COCO) dataset [18] which has 80 categories. The deep learning-based vehicle classifier broke through the limitation of traditional classifier and made a tremendous improvement in mAP scores. This

section will introduce several famous vehicle classifier and related algorithms.

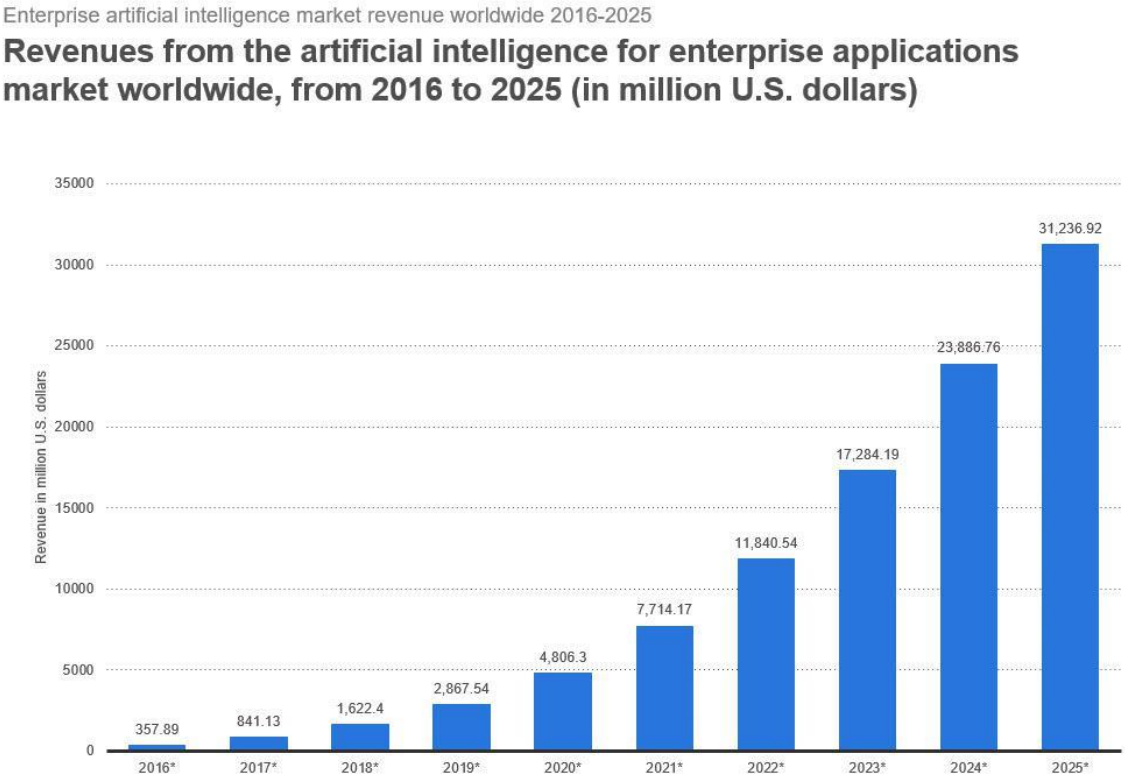


Figure 2.1: The trend chart of AI statistics is drawn to represent the trends [19].

2.1.1 Overview of Recent Trend

Recent vehicle classifier can be categorized into two branches, One-Stage classifier, and Two-Stage classifier. One-stage classifier makes the region proposal for finding Region Of Interests (ROI) and vehicle classification at the same time, while two-stage classifier first find the ROI and feed those ROIs into the classification process. There is a trade-off between one-stage classifier and two-stage classifier, one-stage classifier tends to have lower computational cost and thus have faster processing speed, while two-stage classifier tend to have higher performance since the region proposal part is separated, which leads to better ROI qualities, result in better classification results and tighter bounding boxes.

R-CNN family is the frontrunner in Two-Stage vehicle classifier, which was first proposed by [19], this version first does region proposal done by selective search and then feeds those ROIs into a convolutional neural network individually for classification and bounding box regression. Because those ROIs actually partly cover the same pixels with each other and result in duplicate computation and huge computational cost, to solve this problem, researchers

proposed Fast R-CNN, they feed an image into the CNN only one time and by using ROI pooling they resize the feature map of ROIs into a fixed-size feature and then further feed them into a fully-connected network to do classification and bounding box regression. They reduced the training and testing time by around ten times compared to R-CNN. But there is still a problem in the pipeline of Fast R-CNN, the region proposal part is still based on the traditional method (i.e., selective search). Thus, their model is unable to optimize in an end-to-end manner. Faster R-CNN [20] addressed this problem and proposed Region Proposal Network (RPN), RPN is trained by fitting the default anchor boxes to ground-truth bounding boxes, RPN substituted selective search to make region proposal and finally made the training pipeline end-to-end.

For One-Stage classifier, the first appearance is OverFeat [21], they combined the localization and classification within a single network, SSD proposed a method doing region proposal with the concept of anchor box used in RPN in Faster R-CNN along with doing bounding box regression and classification at the same time, SSD reached competitive performance compared to two-stage classifier, DSSD is the enhanced version of SSD which introduced the deconvolutional process in their network. You Only Look Once version-3 (YOLOv3) [22] also obtained remarkable results by utilizing multi-scale feature maps inside the network. RetinaNet [23] introduced Focal Loss to handle the unbalanced sample problem, Focal Loss will be explained in section 2.3. CornerNet [24] is a key-point based method that finds the top-left and bottom-right points of a vehicle and further performs bounding box refining. Researchers in [25] proposed an anchor-free method (region proposal without anchor boxes), which can be applied to one-stage classifier, with the combination of anchor-based method and anchor-free method, they achieved the state-of-the-art vehicle detection results. Table 2.1 lists the components in the recent vehicle classifier.

Table 2.1: Components of Vehicle Pose Estimation with License Plate Localization

Vehicle Pose Estimation with License Plate Localization			
Classification Paradigms	Feature Extraction	Proposal Generation	Backbone Architecture
One-Stage	Image Pyramid	Anchor-Based	VGG16 [26]
Two-Stage	Detection Pyramid Integrated Features	Anchor-Free	ResNet [27]
	Feature Pyramid		Hourglass Net [28]

2.1.2 Feature Extraction

To generate final vehicle detection results, an encoding method for image information is necessary, and this is often called feature extraction in the vehicle detection task, lots of convolutional neural network architectures are designed for this purpose, the DCNN architecture used for feature extraction is then viewed as backbone architecture.

On top of the backbone architecture, how to utilize the feature map generated in different inner layers of the network has been studied widely in recent years. In the convolutional neural network, a sequence of down-sampling process will be performed, in this process, early layers with high resolution often have richer spatial information and weaker semantic information, deeper layers with lower resolution often have less spatial information but more semantic information. This leads to a property that early layers are good at detecting small vehicles but lack of the ability to recognize and classify vehicle categories, on the other hand, deeper layers are short of finding small vehicles but more capable of categorizing vehicles and more robust on different translation, illumination, and transformation of a vehicle.

Vehicle classifier like Faster R-CNN and YOLOv3, they detect the vehicle by only utilizing the final output feature map, to overcome the spatial information-lacking problem, a method is to intuitively, train one classifier with different scales of images or to train several classifiers with each classifier handling different sizes of images and then combine the results. This process is called Image Pyramid and shown at the top left in Figure 2.2. This method could be computationally expensive since we need to feed the image several times at the training and testing stage.

Later, SSD used Detection Pyramid, which is shown at the bottom left in Figure 2.2. They utilized the inner layer's feature map in the backbone DCNN and then performed prediction at different resolutions. Since the early layers contain spatial information for grabbing small vehicles, it makes the multi-scale vehicle detection possible within a single network. However, the problem is, we still need to do several times of detections and combine the results.

Stacked Hourglass Network [28] and U-Net are designed under another concept, Integrated Features, shown at the top right in Figure 2.2. They fuse the interior features and perform the prediction on a single output map taking advantage of utilizing multi-scale features, and only single detection is needed. In our research, the model we used is based on Hourglass Network. The base architecture of the Hourglass Network is described in section 2.2, and our model design is in section 3.1.1.

Combining the concept of both Detection Pyramid and Integrated Features, Feature Pyramid [28] was proposed, which is at the bottom right in Figure 2.2. They first fuse the features extracted from different levels of layers, perform the detection in each level, and then further combine the detection results. RetinaNet and YOLOv3 adopted Feature Pyramid and obtained promising results. Although Feature Pyramid is experimented to yield the best detection results, considering the trade-off between performance and inference time, we chose Integrated Features as our final feature extraction choice since the applications of license plate detection task often prefer real-time processing.

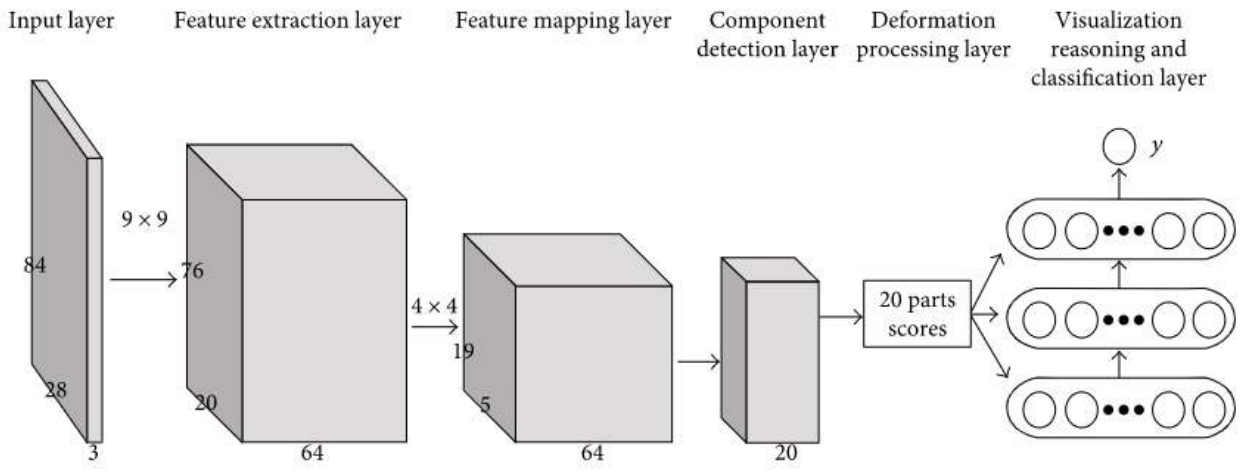


Figure 2.2: Different feature extraction methods [28].

2.1.3 Region Proposal Methods

Anchor-based: Before the era of deep learning, Selective Search [29] was a simple and reliable region proposal method. However, **the critical drawback is the disability for a model to train and optimize end-to-end when applying Selective Search. In the case of R-CNN, they originally used Selective Search as their region proposal method and introduced Region Proposal Network (RPN) later in Faster R-CNN, making the region proposal process be a neural network as well. In the training process of RPN, several manually designed default anchor boxes will be regressed to ground-truth bounding boxes with refining parameters, after the RPN has been appropriately trained, at inference time, the anchor boxes with high vehicle-existing probability will be treated as a potential region proposal, the others will be treated as background.**

Anchor-based method almost dominates the region proposal part among recent vehicle classifier because of its ability to detect vehicles in different aspect ratios and sizes, proposing multiple vehicles within a single grid is also possible by applying anchor-based method. But the designing and choosing of anchor box's aspect ratios and sizes are all manually decided, as the process is done in a heuristic manner, it might be a limitation for a network to bring out their best potential. Some efforts have been made to choose better anchor box, In [30] they used k-means to cluster the training data for obtaining anchor boxes with more preferable aspect ratios, in [31] for detecting vehicle's front and rear also introduced k-means clustering to decide the default anchor boxes.

Although the refined method for defining anchor box does not count on heuristic design anymore, the large variety of aspect ratios and sizes of vehicles around the world are not even possible contained by a limited amount of anchor boxes, so here we state the potential problems of anchor-based method:

1. Anchor box designed by referencing training data might not be compatible with real cases.
2. The concept of anchor box is not natural since it is not possible to represent the abundant vehicle categories in the world with a limited amount of anchor boxes.

Anchor-free: Early one-stage vehicle classifier like Overfeat [32] and YOLOv3, their region proposal stage was anchor-free design, YOLO treated the bounding box refining as a regression

problem, so the region proposal and classification tasks are all done by a series of DCNN followed by fully-connected layers.

After that, anchor-free method lost its attention for several years after RPN [33] brought out their anchor-based method. Nevertheless, recently, CornerNet [34] is a trained weighted network which works on the corners of image and CenterNet [34] is a trained weighted network which works on the center of image put efforts on constructing anchor-free vehicle classifier. CornerNet tried to find the top-left and bottom-right corners of a vehicle instead of finding bounding box, they proposed a novel pooling method called Corner pooling, which encodes the corner message of a vehicle and is claimed to be the critical stage for the success of CornerNet. CenterNet first predicted the bounding boxes by pairs of corners and then predicted the center probabilities for a vehicle to exist. It is based on extracting the information of the center and the corners of a vehicle. Both of the anchor-free vehicle classifier obtained promising results. CenterNet presents their best Average Precision (AP) on MS COCO dataset as 47.0, surpassing all of the previous one-stage vehicle classifier and most of the previous two-stage classifier.

Inspired by the impressive results of CornerNet and CenterNet, our region proposal method is designed under the concept of anchor-free method and is described detailed in section 3.1.2 and 3.1.3.

2.2 STACKED HOURGLASS NETWORK

Stacked Hourglass Network was first designed for human pose estimation; it is a feature extraction backbone with several Hourglass Networks connected together. A single Hourglass Network is shown in Figure 2.3; the architecture of Hourglass Network is a sequence of down-sampling DCNN followed by a sequence of up-sampling DCNN. To utilize the feature with rich spatial information (shallow part of DCNN) and feature with rich semantic information (deeper part of DCNN), there're lateral connections that merge the early features with latter features, it's the central concept of Hourglass Network and make it suitable for vehicle detection tasks since the spatial information often disappears in the latter part of a simple DCNN architecture.

Each block (except the block with * mark) in Figure 2.3 refers to a residual block shown in Figure 2.4. The residual block was first proposed by [35], a skip connection from input layer directly to output layer was added, it was designed to address the problem of gradient vanishing

problem when training a deep neural network. The network goes deeper, information from shallower layer might disappear due to the gradient-based back-propagation, residual block makes it possible to train a network as deep we want. As we can see in the input and output of the residual block, the channel amount (kernel amount) is basically 256 in the entire Hourglass Network. Inside the residual block, there are two DCNNs with 128 channels with kernel sizes 1 and 3, followed by a DCNN with 256 channels with kernel size 1. Stacked Hourglass Network was used for feature extraction backbone network in recent vehicle detection researches like CornerNet and CenterNet.

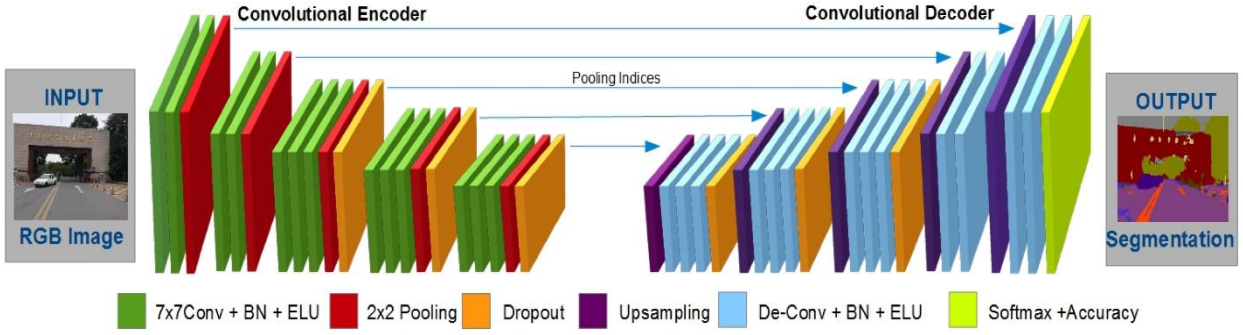


Figure 2.3: Hourglass Network with block refers to simple feature addition; other blocks refer to residual blocks [35].

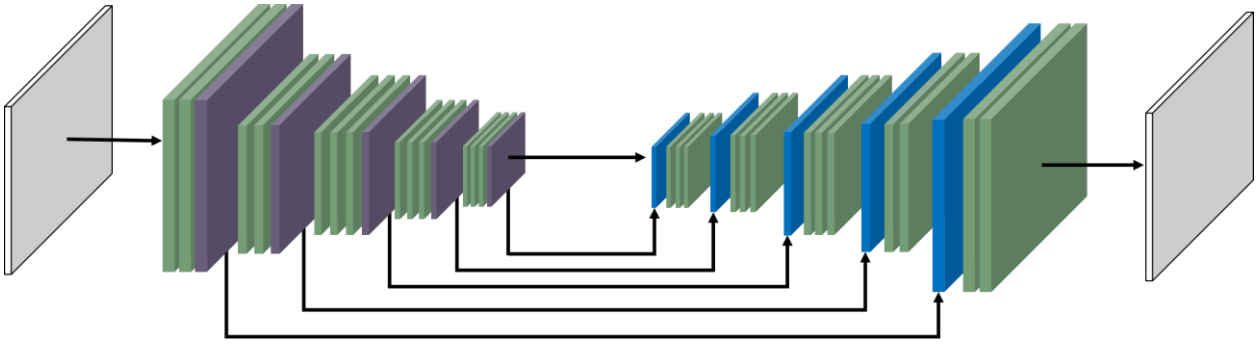


Figure 2.4: Residual block used in Hourglass Network [35].

2.3 FOCAL LOSS

Focal Loss [36] is an optimization strategy aiming at handling imbalanced samples. Since in the training process of a vehicle classifier, ground-truth positive labels are always less than ground-truth negative labels with a huge gap. To prevent the loss created by the ground-truth negative labels overwhelming the training stage, methods for handling this problem is essential and often called hard negative mining.

Controlling the data feeding during the training process like Online Hard Example Mining (OHEM) [37], and forcing the ground-truth positive and ground-truth negative ratio to be 1:3 in SSD are some of the methods for hard negative mining.

Instead of directly handling the training samples fed into the model, Focal Loss tried to handle this problem by controlling and giving different weights to the loss created by easy-samples and hard-samples, and the name 'Focal' refers to 'Focusing' on hard-samples.

$$FL(p_t) = -\alpha(1-p_t)^{\gamma} log(p_t) \tag{2.1}$$

$$\alpha_t = \begin{cases} \alpha & if\ y = 1 \\ 1-\alpha & otherwise \end{cases} \tag{2.2}$$

$$p_t = \begin{cases} p & if\ y = 1 \\ 1-p & otherwise \end{cases} \tag{2.3}$$

FL is the formula of Focal Loss; it can be considered as an extended form of Cross-Entropy loss. A factor αt is added for controlling the loss weight, which ranges between 0 and 1, y is relevant to the ground-truth label, for positive samples, y equals to 1, otherwise equals to 0. Say we set αt to 0.9, the weight contribution ratio between ground-truth positive samples and ground-truth negative samples will then become 9:1 (0.9:0.1). Adding αt is able to adjust the contribution between positive samples and negative ones, but not considering the easy cases and hard cases.

To make the model focus on the hard cases which are more difficult to learn, the term $(1-p_t)^{\gamma}$ is added. p_t is the vehicle-probability output of a vehicle classifier, it will be close to 1 when it's true-positive or true-negative and close to 0 when it's false-positive or false-negative. γ is a factor to control the weight decay for easy-samples, $(1-p_t)$ will be close to 0 if the model makes true predictions while close to 0 on the other hand, with γ value larger than zero, the loss caused by true predictions (easy-samples in the early stage of learning) will be shrunk and the loss caused by false predictions will remain the same. When γ equals zero, then the loss function is the same as Cross-Entropy loss. The difference between Focal Loss with various values of γ and Cross-Entropy loss is mentioned in the original paper and shown in Figure 2.5.

In our work, we used extended forms of Focal Loss in the localization loss and classification loss and described them in section 3.2.1.

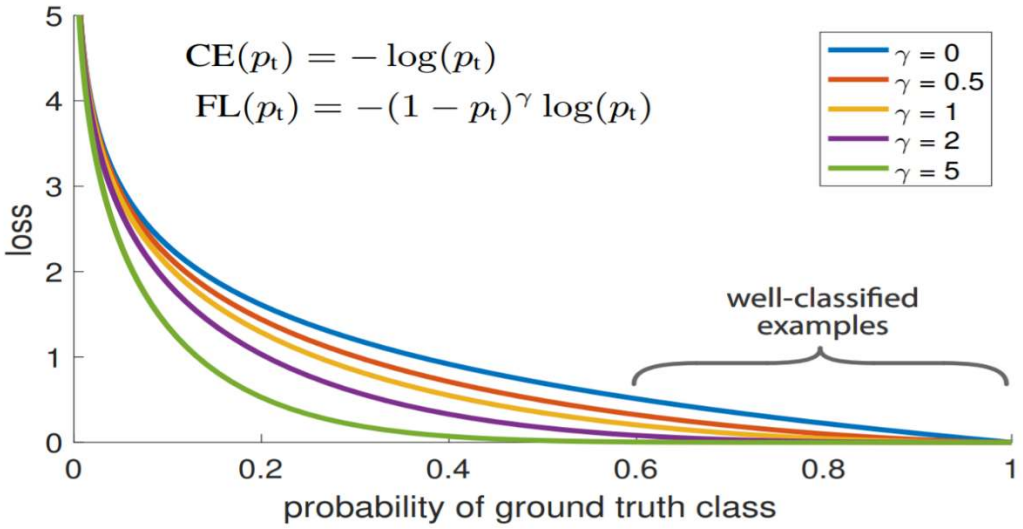


Figure 2.5: Focal Loss the Cross-Entropy loss is the curve γ= 0. The loss with larger $\boldsymbol{p_t}$ value diminished eminently with different levels depending on the value of γ [36].

2.4 NON-MAXIMUM SUPPRESSION (NMS)

In the test stage of a vehicle classifier, the output vehicle probability larger than a decided threshold will be considered a positive detection result. There will be a massive amount of duplicated bounding boxes for a single vehicle, to eliminate the duplicated ones and keep the most-likely one, we often perform Non-Maximum Suppression (NMS).

$$Score_B = \begin{cases} Score_B & if\ IoU(B,M) < \Omega_{test} \\ 0 & if\ IoU(B,M) > \Omega_{test} \end{cases} \quad (2.4)$$

Equation (2.4) shows the process of NMS, NMS first arranges all of the detection results according to their probability score (Score B) of having a vehicle in descending order. Let 'M' be the result with the highest score, compare the results B having less score than M, if the Intersection over Union (IoU) between M's bounding box and B's bounding box is greater than a specific threshold Ω test, then the score of B will be set to zero, or to say remove the detection result of B, else if the IoU is smaller than Ω test, B will remain there as performed in [38]. This process performs iteratively until the detection result with the lowest score has been checked. Figure 2.6 gives an example of bounding box removal in NMS pipeline.

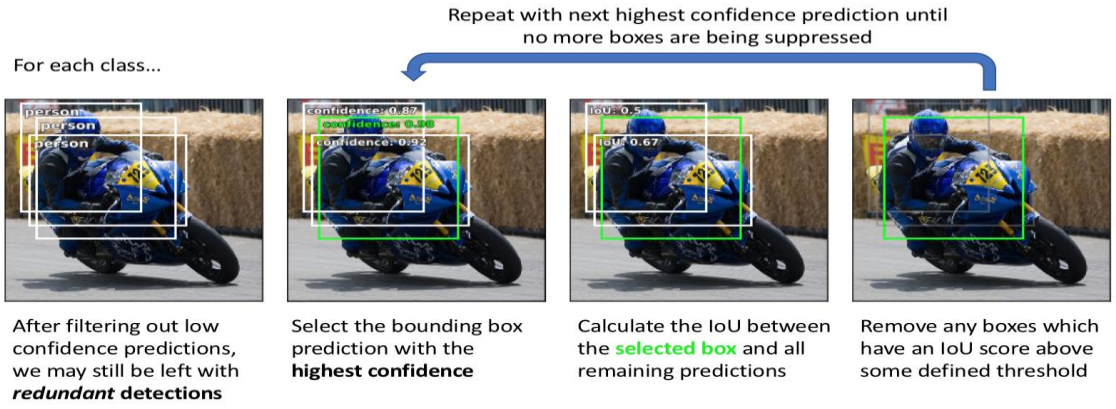


Figure 2.6**:** Illustration of Non-Maximum Suppression (NMS) with all four steps of NMS pipeline, the bounding box which best fits the vehicle in detector output has the highest score [38].

3. METHODOLOGY

There are mainly four issues we would like to tackle, now we list the corresponding methods and works for each problem and their relevant sections:

1. Lack of detection ability for tilted and oblique license plates. To make our model able to locate the license plate accurately, we need a strong feature extraction backbone, and a proper design for indicating the license plates with diverse transformations under different scenarios explained in 3.1. The loss function designed for optimizing with back-propagation is described in section 3.2.

2. Loss of contextual information while dealing with more general license plate detection. Contextual information is defined as the relationship between the vehicle and the license plate in our work. For obtaining the vehicle's pose information when a license plate is detected, our method is to find the front part and rear part of a vehicle, with the information of car's front and rear, we can obtain the contextual information more precisely as discussed in section 1.4. As we will demonstrate in section 3.1, our model can receive either ground-truth bounding quadrilaterals or bounding rectangles as our training data while not limit to only bounding rectangles training, and it is due to our anchor-free design for finding four vertices of ROI individually, the corresponding method for optimizing such a design is mentioned in section 3.2. In 3.3, we give a view of the training method we applied during our training, including the data augmentation and the training strategy refined at times by observing the learning condition of our model.

3.1 DEEP CONVOLUTIONAL NEURAL NETWORK

The proposed model is called DCNN which is a one-stage and anchor-free deep learning classifier. The whole model is shown in Figure 3.1. The width and height of an input RGB image will be first downscaled to 1/4 and pass through two stacks of Hourglass Network, until here is the backbone feature extraction part. By utilizing the obtained features, three parallel heads will then handle localization, region regression, and classification individually. The right side of Figure 3.1 gives an output example of our model, we can see that the license plate region has been found, and the front-rear region of the owner car is also given along with its pose information.

This section includes the backbone feature extraction network design, and the head network architecture with its function and design inspiration, which is done in one-stage manner. The last part of this section will be a brief discussion for anchor-free method.

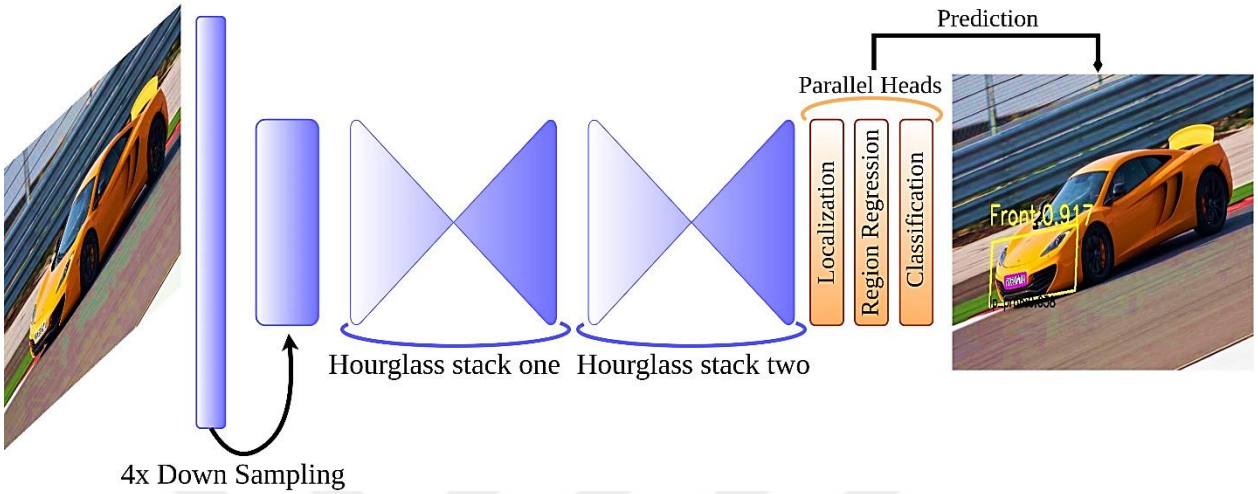


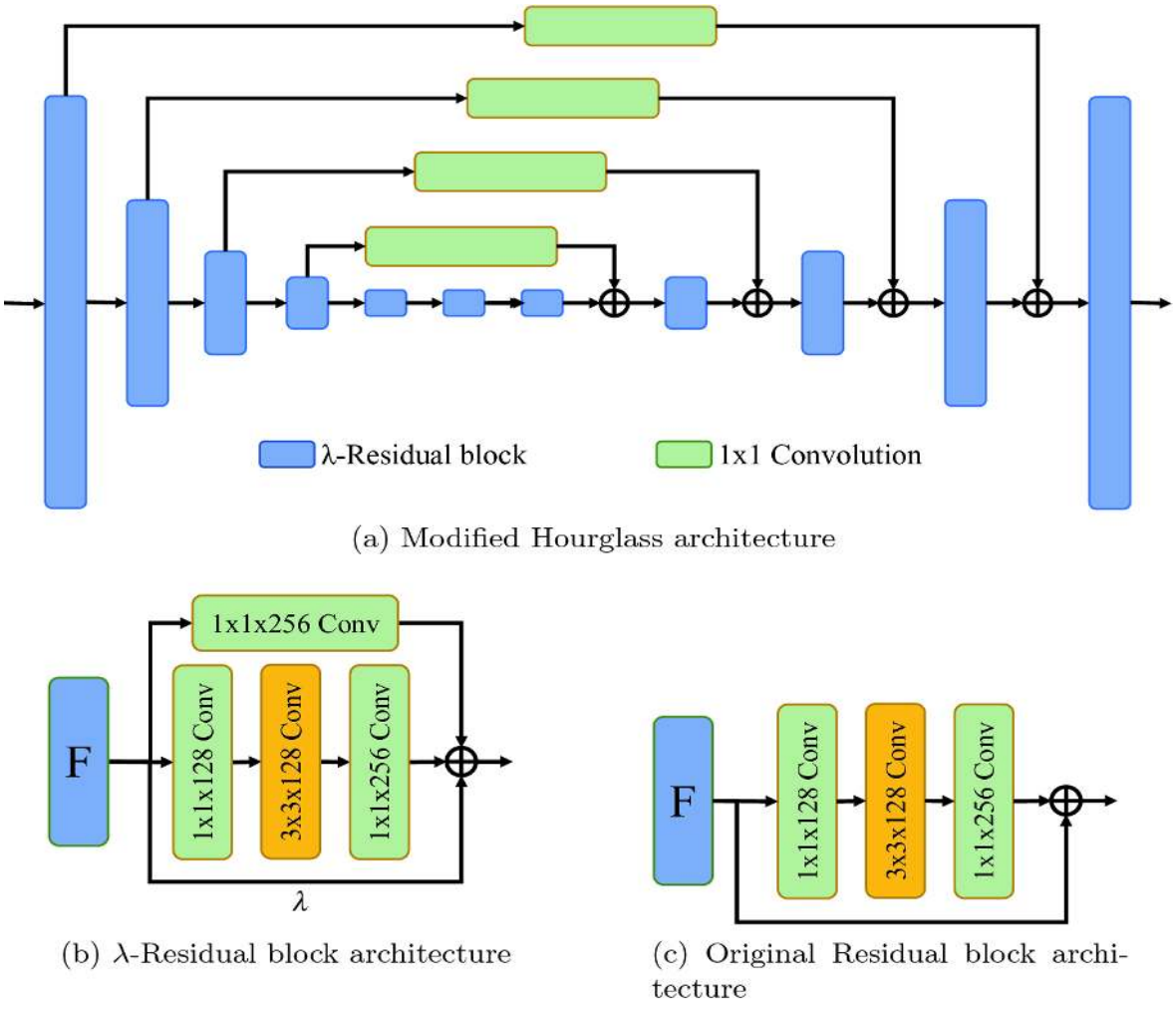
Figure 3.1: The whole pipeline of proposed model [38].

3.1.1 Backbone Network Design

As discussed in section 2.1.1, a one-stage classifier has a trade-off between speed and performance, and the performance will descend significantly especially for small vehicles. License plates, in general situations, are small vehicles (imaging the scale difference between license plates and vehicles), so the feature extraction ability of the backbone network architecture becomes considerable, the spatial information needs to be fruitful to avoid missing detections for small vehicles. Inspired by the adoption of Hourglass Network in recent one-stage detectors, we introduced Hourglass Network as our backbone architecture and found it performed much better than a normal straight forward DCNN, a comparison between their performance.

The network design is given in Figure 3.2, our training input size is fixed in 256x256, after two down-sampling processes (a convolution layer and a max-pooling layer with stride 2), the image with its scale of 1/4 width and height will then be fed into a two-stack Hour-glass Network. The layers in Figure 3.2 without special mention are all Rectified Linear (ReLU) activation function, and the layers in trapezoids are down-sampled or up-sampled processes by a factor of 2.

The right side of Figure 3.2 is the architecture of a single Hourglass Network, the Residual block (ResBlock) architecture is shown in Figure 3.3. After passing through the first Hourglass Network, following the original design of stacked Hourglass Network, we add two parallel DCNN branches with linear activation functions and do the element-wise addition with the features before feeding into Hourglass Network in a residual manner. The feature map after performing element-wise addition will then pass through another Hourglass Network with the same architecture and yield a final output with size 64x64.



(a) Modified Hourglass architecture

(b) λ-Residual block architecture

(c) Original Residual block architecture

Figure 3.2: Backbone two-stack Hourglass Network [39].

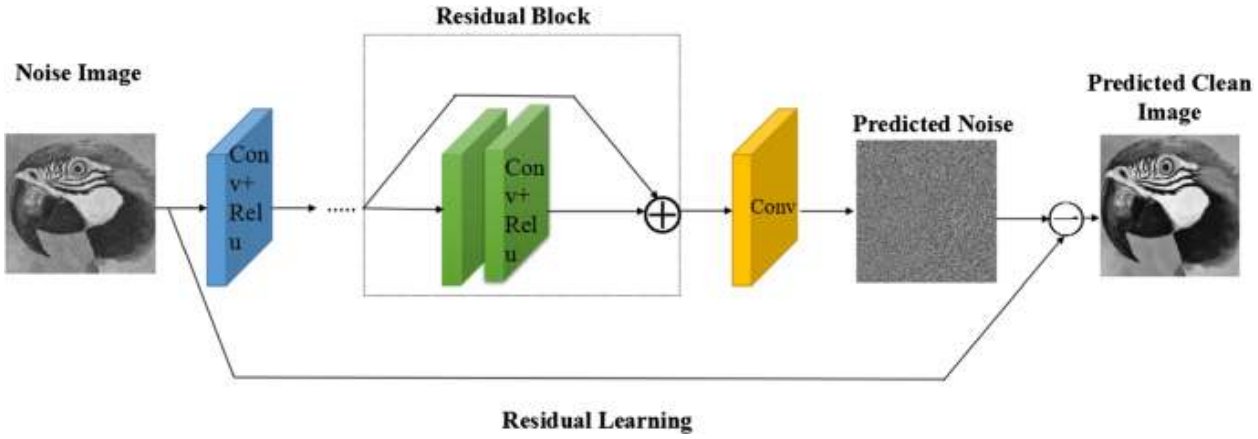


Figure 3.3: Residual block [40].

3.1.2 Head Network Design

Beyond the backbone network, we designed four parallel DCNNs as our

..... head networks, each of them handles different tasks, Figure 3.4 gives a clear comprehension of the architecture, and we will illustrate each of the head networks in this section.

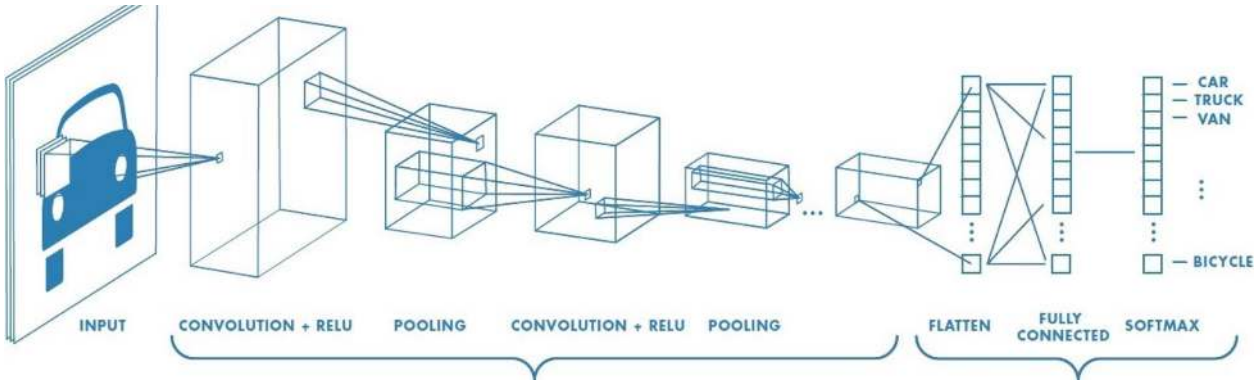


Figure 3.4: Head network, there are four parallel DCNN branches, the top localization branch, the middle two region regression branches, and the final classification branch [41].

1. **Localization for vehicle position and license plate:** The localization head is done by a deep convolutional neural network with filter size 3x3, the output will be a feature map of size 64x64x1, for each pixel, there is only one output channel which indicates the probability of containing a license plate. Since we did not include the background class in our design,

it serves as a single-class problem; the sigmoid activation function fits our requirement and became our choice for activation function.

2. **Region regression for pose estimation and license plate localization:** The region regression part is done by a convolutional neural network with filter size 3x3, and the output feature map has a size of 64x64x8, there are eight output channels for each pixel, the output channels handle the region proposal for vehicle pose estimation and license plates localization. Figure 3.5 explains how these values work with region proposal, first, we will have four pairs of unit vectors in the same arrangement of four quadrants, r_x and r_y then serve as scalars to expand these unit vectors, after expanding a pair of unit vectors, do the vector addition of the horizontal vector and vertical vector in the pair to obtain a destination point. After performing the same procedure on the four pairs of unit vectors, we will obtain four points, and these points will then be the vertices of a quadrilateral, which indicate the region of a license plate. Since we need the expanding factor for unit vectors, exact scalars must be obtained, so we used linear activation function (equivalent to no activation function) for the DCNN layer.

3. **Region regression for front-rear:** The region regression for the car's pose (front-rear) uses exactly the same method used in region regression for license plate, but the vertices now indicate the front-rear region of a car.

4. **Classification for front-rear:** As a multi-class classification task, there are three classes here, front class, rear class, and background class. The classification process is done by a convolutional neural network with filter size 3x3, and the output feature map has a size of 64x64x3, each channel in each pixel represents the probability for front class, rear class, or background class, respectively. Here we used softmax activation function for its compatibility for multi-class classification.

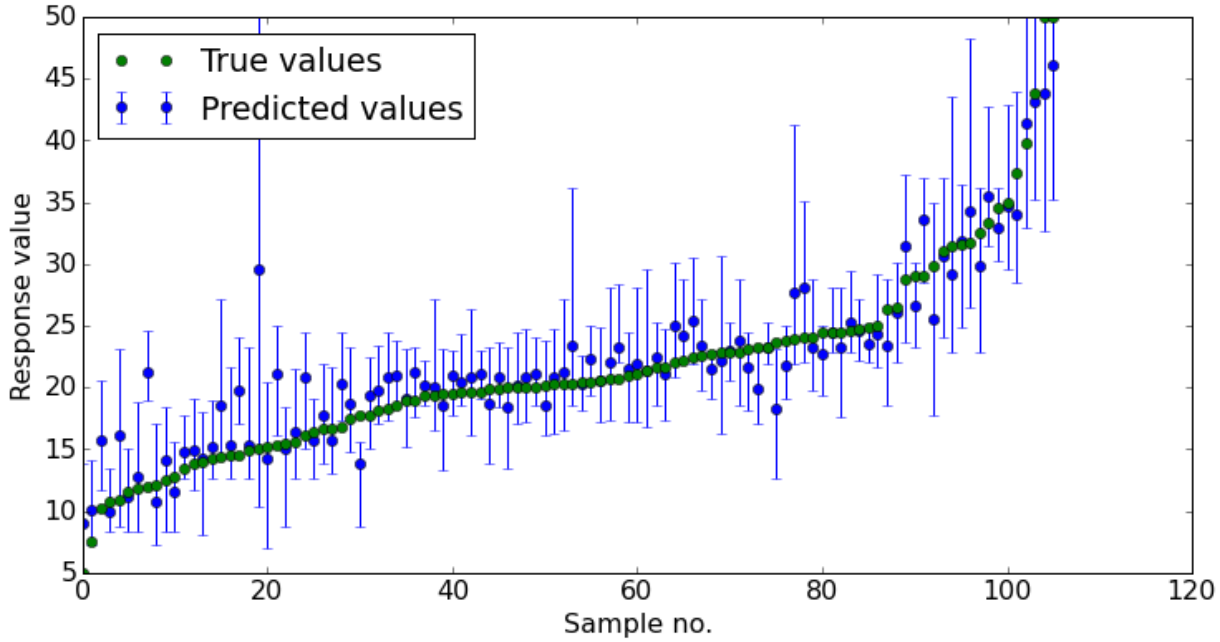


Figure 3.5: Region regression method [42].

3.1.3 Anchor-Free Method

We used anchor-free design for our head network, it can be observed that no default anchor boxes are needed in the region proposal process, it means that different from anchor-based vehicle detectors, during our training process, there are no heuristic decisions, all we need to do is to let our model naturally learn the ability of detection. Apart from this, since our model regresses the four vertices of a quadrilateral individually, not only the bounding rectangle, we can also find the exact vertices of a license plate, after performing planar rectification by perspective transform, we can do further OCR on a more precise region.

3.2 OPTIMIZATION STRATEGY

This section explores the optimization method for training our model, including the loss function design for each head network, and the label encoding method which encodes the spatial information into an available format for the loss functions.

3.2.1 Loss Function

The introduction of the loss function is arranged in the same order as the head network design section. For the localization of the license plate, we used Focal Loss as our foundational loss

function. Equation 3.1 gives the localization loss for each pixel (m, n) in the output feature map. The first two terms are the Focal Loss part, where $P_{true,lp}$ is the ground-truth label of the license plate, P_{lp} is the predicted probability. We set α=0.75 for amplifying the loss influence of positive cases and thus balance the ratio between positive cases and negative cases. For the Focal Loss factor, we set γ=2 by following the best result from the released paper [39]. Here, we introduce the contextual information auxiliary training, which is the last term in Equation 3.1. The inspiration is that there will not be a license plate outside the region of a vehicle, so we designed an additional term for this purpose, $P_{true,front\text{-}rear}$ refers to the ground truth label for the areas of vehicle's front part or rear part, which is annotated in our proposed dataset. For typical license plate detection purposes, the license plates are arranged onto the front-rear part of a car, and the additional term gives an extra penalty if a positive license plate detection is made outside the region of a vehicle. We set a factor β=0.5 to balance the loss contribution for avoiding this term overwhelming the entire training process. Figure 3.6 gives an intuition for the difference between adding the contextual information auxiliary term and without adding it. With the aid of this additional term, the model is able to avoid the false positive detection of license plate outside the region of a vehicle.

$$L_{localization}(m,n) = -P_{true,lp}^{m,n} \,.\, \alpha \,.\, \left(1 - P_{lp}^{m,n}\right)^{\gamma} .\log(P_{lp}^{m,n}) - \left(1 - P_{lp}^{m,n}\right)^{\gamma} .(1 - \alpha) .\left(P_{lp}^{m,n}\right)^{\gamma} .\log(1 - P_{lp}^{m,n}) - \left(1 - P_{true,front-rear}^{m,n}\right) .\beta .(P_{lp}^{m,n}) .\log(1 - P_{lp}^{m,n}) \quad (3.1)$$

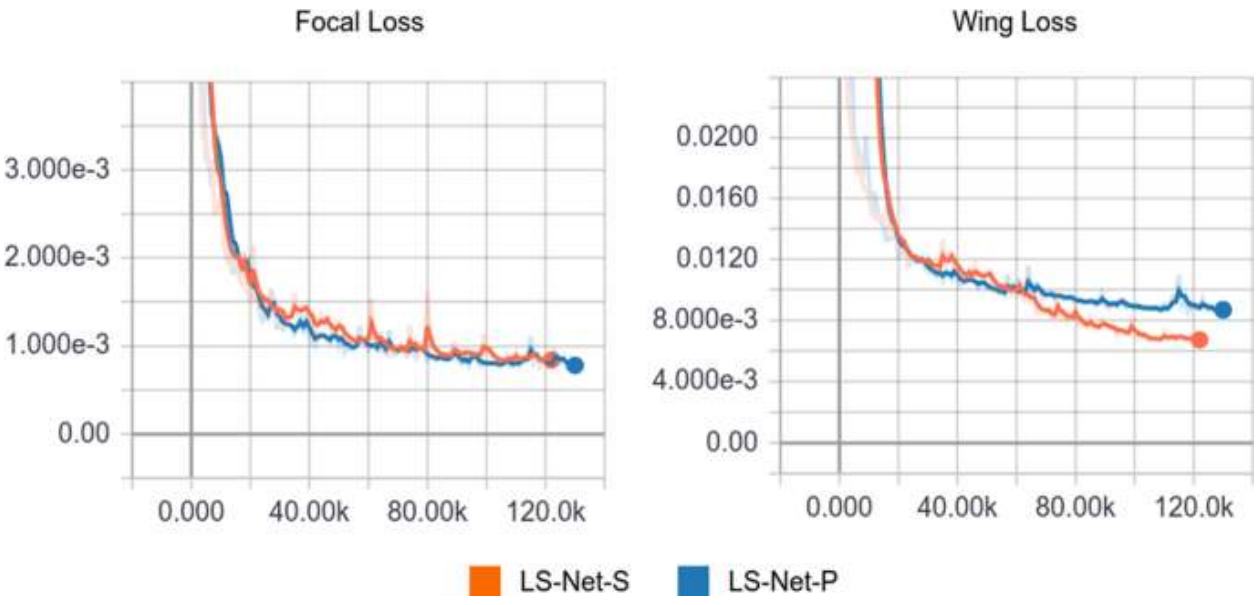


Figure 3.6: Comparison between Focal Loss with and without contextual information auxiliary [43].

The second part of the loss function is the region regression part. We used L1 loss for this purpose. Equation 3.2 shows the loss of a single pixel (m, n) in the output feature map. $G_{Ti,x}$ and $G_{Ti,y}$ refer to the x and y coordinate of the four ground truth vertices, i=1~4 from bottom right and clock-wise. $v_{i,x}$ and $v_{i,y}$ refer to the basic vectors mentioned in section 3.1.2, $T_{i,x}$ and $T_{i,y}$ are the output values of the region regression network, the multiplication of v and T then yields the final prediction coordinates. The loss will be the summation of the L1 losses of the four vertices. A factor 1/ε was added to normalize the loss value since the L1 region regression loss will be relatively large compared to localization loss and classification loss due to the linear activation function. For the small vehicle, license plate, we set ε=1, and for the car's front-rear, considering the out-feature map size of our model (64x64), we set ε=32.

$$L_{region}(m,n) = \frac{1}{\varepsilon} \cdot \sum_{i=1}^{4} \left| GT_{i,x}^{m,n} - V_{i,y}^{m,n} \cdot T_{i,y}^{m,n} \right|$$

(3.2)

$(GT_{i,x}^{m,n}, GT_{i,y}^{m,n})$= ground truth coordinate of vertex_i

$$\left[V_x^{m,n}, V_y^{m,n}\right] = \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ -1 & -1 \\ 1 & -1 \end{bmatrix}$$

$$\left[T_x^{m,n}, T_y^{m,n}\right] = \begin{bmatrix} r_{x1} & r_{x1} \\ r_{x2} & r_{y2} \\ r_{x3} & r_{y3} \\ r_{x4} & r_{y4} \end{bmatrix}$$

The last part is the classification loss, which is shown in equation 3.3. We used the multi-class Focal Loss. $P_{true,front}$, $P_{true,rear}$, and $P_{true,BG}$ refer to the ground-truth positive cases for front class, rear class, and background, respectively. P_{front}, P_{rear}, and P_{BG} refer to the predicted probability. We set γ=2 in the training process.

$$L_{class}(m,n) = -P_{true,front}^{m,n} \cdot \left(1 - P_{front}^{m,n}\right)^{\gamma} \cdot \log\left(P_{front}^{m,n}\right) - P_{true,rear}^{m,n} \cdot \left(1 - P_{rear}^{m,n}\right)^{\gamma} \cdot \log\left(P_{rear}^{m,n}\right) - P_{true,BG}^{m,n} \cdot \left(1 - P_{BG}^{m,n}\right)^{\gamma} \cdot \log\left(P_{BG}^{m,n}\right)$$

(3.3)

Equation 3.4 gives the final loss function. The total loss is the summation of localization, region regression, and classification. We only calculate the region regression loss when the ground-truth label of the pixel is positive; otherwise, it is set to zero. Note that there are two terms of region regression loss, one for license plate and one for car's front-rear region.

$$L_{Total} = \sum_{m=1}^{M}\sum_{n=1}^{N}(L_{localization}(m,n) + P_{true,lp}^{m,n}.L_{region,lp}(m,n)\ ) + P_{true,front-rear}^{m,n}.L_{region,front-rear}(m,n) + L_{class}(m,n) \quad (3.4)$$

3.2.2 Label Encoding

With only the bounding box annotation in our training dataset, we need several label encoding methods to meet the loss function in the optimization process for each head network. Figure 3.7 visualizes the label encoding for a single input image. For simplicity, we will use GT as the abbreviation of Ground-Truth. To clearly describe the type of bounding box, we will use the terms bounding rectangle and bounding quadrilateral in the following content.

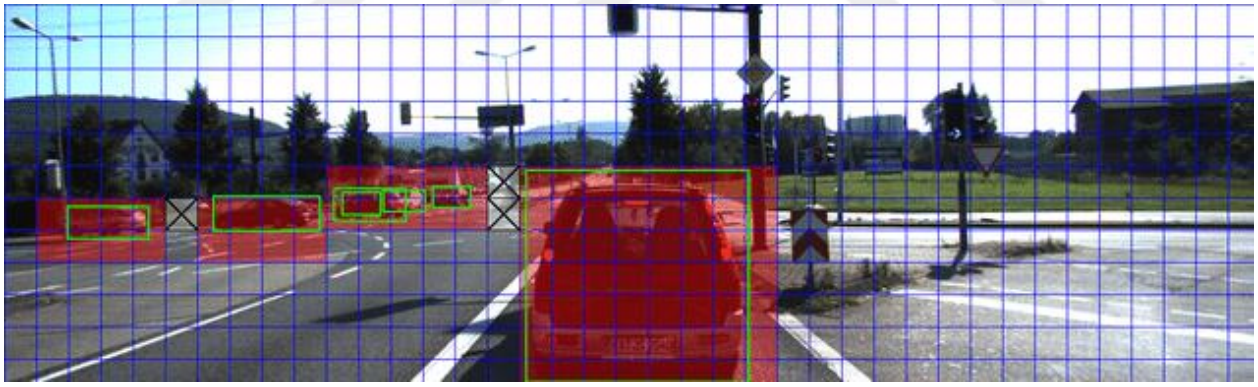


Figure 3.7: Label encoding method [44].

For the localization loss function, we need to give each pixel a label indicating either the pixel is inside a license plate or not, which is P_{true}, l_p in the loss function. We first mapped the GT bounding quadrilateral coordinates to the scale 64x64 to meet the output size of our model. We then define a bounding rectangle having the same size with the GT license plate bounding rectangle (here, we transferred the annotations from bounding quadrilateral format to bounding rectangle format), and then let each pixel inside the GT bounding rectangle be centroid of that bounding rectangle, calculate the IoU value between it and the GT bounding rectangle, if the

value is larger than a threshold, then we assign the pixel value to 1, else assign to 0. As a result, pixels with value 1 means the ground-truth positive. The upper branch of Figure 3.7 shows the encoded label; the black region contains the pixels with value 1. We set the threshold to 0.7 for the localization label encoding.

3.3 TRAINING DETAILS

The purpose of this section is to describe the whole construction of our training process, including the online data augmentation method, the transfer learning method used for avoiding unstable training state, and the adjustment of the hyper-parameters for different periods of iterations.

3.3.1 Data Augmentation

We used online data augmentation. Before the training images were fed into the model, several augmentation methods were taken randomly. These methods are listed in Table 3.1, the Chance column in the table with value 50% means the augmentation was not taken for every time, but only half of the chance. Each image performed each of the methods for one time with random order.

Online data augmentation preserves the diversity of training data. In every augmentation method, we also used random parameters, meaning that after performing all of the methods, a single original image will be transformed into countless augmented images, this expands the training data scale and prevents the model from overfitting too early, Figure 3.8 gives some augmented data examples of a single image.

Table 3.1: Augmentation for Training Data

Augmentation type	Parameter	Chance
Horizontal scale	0.5 ~ 1.0	50%
Vertical scale	0.5 ~ 1.0	50%
Shear angle	-90 ° ~ +90 °	50%
Rotation angle	-25°~+25°	50%
Perspective transform	Standard deviation: 0.05 ~ 0.1	50%
Flip	Only horizontal flip	50%
Hue and Saturation	-50 ~ +50 (in scale of 255)	100%
Overall scale	0.2 ~ 1.0	100%

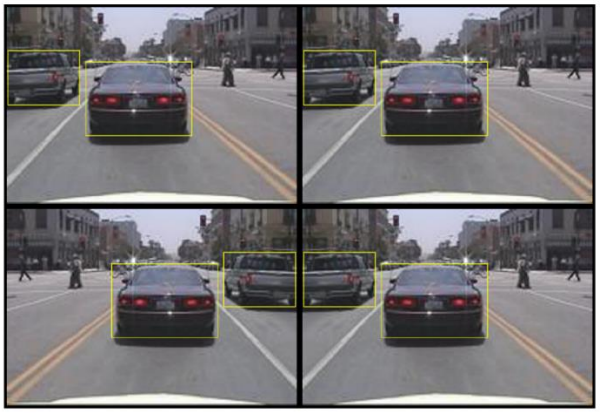


Figure 3.8: Samples of augmented data [45].

3.3.2 Step by Step Transfer Learning

There are three functional head networks in our design, and thus the end-to-end and simultaneous training leads to a potential training instability. If the model fails to converge, it is quite hard to trace which head design is raising a problem. Thus, in our design process, we added each the functional head step-by-step, making sure the single head design idea is working correctly and then expanded the model with the rest functional heads. We will explain the model expanding process in this section.

The model expanding process can also be viewed as a transfer learning process. In the initial design of our model, we only had the localization head and region regression head for license plate detection, after confirming that our novel anchor-free method for region regression is working as we expected, we then added the heads for car pose detection. We loaded the weights we trained, both the weights in the backbone architecture and the license plate heads, and then trained the newly added car pose heads. After the car's pose detection has been trained to an acceptable level, we then added the last puzzle in our model, the head for car pose classification.

Our final model training started from an initial weight which we had already trained for about 110000 iterations in the model without classification head, so the total training iterations will be around 780000 including the transfer learning part in DCNN. Transfer learning made the DCNN model start at a smart point and made it easier to optimize all of the functional heads simultaneously.

4. RESULTS

This section will first give information about the environment and the data we used in our simulation (4.1). Then we show the output result of our model in (4.2), including quantitative evaluation on the benchmark dataset and qualitative visualization results. An analysis of the model learning state will be given in section 4.3.

As mentioned in section 2.4, it is necessary to have a post-processing method for eliminating a large amount of duplicated detection results. All of our final license plate detection results are filtered by a particular NMS method counting on contextual comprehension. Our model will detect a license plate along with the car's front-rear area, and we assume that only a single license plate will appear in that front-rear region, so we performed NMS on the front-rear region instead of the license plate region. We set the IoU threshold = 0.1, meaning that if there is a slightly overlapped region for two front-rear regions, the license plate detection with a lower confidence score will be removed. We found this method to make the detection results cleaner and more reliable, and thus we applied this method in all of our testing processes.

4.1 ENVIRONMENT AND DATA CURATION

This section first describes the hardware and software in our work. Second, for dataset explanation, it includes choosing appropriate images, annotation demand and criteria, and the prospect of training results. We first describe the data we collected for training and then describe the data served as a benchmark to evaluate our model and compare it with other existing models.

4.1.1 simulation Environment

The training and testing process were on operating system Ubuntu 16.04 LTS with CPU Intel i7-6500 3.20GHz, GPU GeForce GTX1080, DRAM 16GB. We built the system with python 2.7, and the deep learning frameworks are Tensorflow and Keras [46]. The codes are all available online. We also made the codes available for Windows and python 3.7. Table 4.1 lists the configuration we set during the training process.

4.1.2 Training Dataset

Our training data contains three components:

- Vehicle images from different nations with various camera views
- Annotation of vehicle pose estimation and license plate localization in either bounding rectangles or bounding quadrilaterals
- Annotation of vehicle's front-rear part in either bounding rectangles or bounding quadrilaterals with front-rear class labels

The camera views differed from benchmark dataset, the vehicles are most in their frontal view, the dataset's license plate annotations are bounding rectangles, in order to eliminate the uncertainty for our model to learn the proper feature of vehicle poses and license plates, license plates with slight rotations were removed in advance since the rectangle bounding boxes can't perfectly describe the license plate. In the dataset, the license plate annotations are in vertices, so the license plate can be fully described even the car is in an oblique view or under some rotations. The original dataset contained 199,993 images, we first chose 2333 images randomly and further chose 1346 tilted or rotated images with a high level of horizontal or vertical rotational degree. Lastly, the images in the dataset were shot under extremely oblique views but with visible characters on the license plate. Table 4.1 lists the overall dataset information.

For the weather condition in the datasets, most of the pictures are taken under clear weather, some of them are rainy or under low level of foggy weather. Images both in the day-time and night are included, while the license plates are all clearly visible.

For the vehicle pose estimation and license plate annotation, the dataset with all bounding rectangle annotations with top-left point and bottom-right point performed and also bounding quadrilateral annotations.

For the front-rear annotations, since none of the LISA dataset gave annotations for the vehicle's front and rear information, we manually annotated all of the images in the datasets mentioned above. Our annotations include

1. The bounding rectangles or bounding quadrilaterals of vehicles front-rear region.

2. The pose information of the vehicle (front or rear).

The reason for the existence of both bounding rectangles and bounding quadrilaterals is that, for LISA dataset, we used bounding rectangles since the view is not that oblique, so bounding rectangles are enough for describing the front-rear part regions. With respect to the dataset, we

used bounding quadrilaterals since the vehicles are in a high level of oblique angles, bounding quadrilaterals then illustrate the front-rear regions much better than simple bounding rectangles. We used open-source annotation tools [47] to do the annotations.

Our final training dataset includes 10824 of images with various nations and camera views to meet the purpose of training a robust license plate detector to handle the diversity of real-world scenes.

Table 4.1: Composition of Training Dataset

Dataset	Number of Images	Region	Description	Horizontal and Vertical Degree
LISA	1107	Brazil	Frontal view	
LISA	2605	US	Frontal view	
LISA	2143	Europe	Frontal view	
LISA	2333	China	Variety of views	Horizontal: 0~30 Vertical 0~30 °
LISA	1346	Turkey	Rotated and tilted	Horizontal: 10 -50 Vertical: 10~50°
LISA	1290	Korea	Extremely oblique	Horizontal: 0~60 Vertical: 0 25 °
Total	**10,824**	**Containing a massive variety of scenarios**		

4.1.3 Evaluation for Multiple Vehicles

For evaluating our model, our LISA benchmark dataset that can closely represent the real-world scenes. Proposed LISA dataset [48], which is a sub-dataset of Cars Dataset. LISA includes 200,000 vehicle images with most of the license plates in oblique angles (difficult cases for license plate detection), and the distance of the vehicles ranges from close, medium to far. Weather conditions are most under clear weather in the day-time. We used 10,824 images in the LISA dataset with a single car inside as our validation dataset (excluding the one with multiple vehicles inside for consistency). In the later section, we use this dataset as a benchmark to compare with other existing systems.

Weather conditions are all under clear weather in the daytime, and the distance of the vehicles is also diverse, with very near vehicles and far vehicles with unrecognizable license plates.

This dataset will be used to evaluate the contextual information missing problem mentioned in section 1.3. A comparison between our proposed method and vehicle detection-based license plate detection method will be made on this dataset.

4.2 PERFORMANCE

Model performance evaluation is divided into qualitative visualization and quantitative evaluation. The qualitative visualization part gives several examples of the detection results on the benchmark dataset and heatmaps for the license plate probability and classification ability. The quantitative evaluation part lists the classification accuracy and mAP performance comparison with existing commercial systems.

4.2.1 Visualization Results

Figure 4.1 shows some examples of the classification results on the LISA dataset, the text above the bounding quadrilaterals of car's front-rear gives the classification results, Front, Rear, or unknown for background class. The number followed by class is the output of the Softmax activation function. The license plate probability is also written at the bottom of the bounding quadrilateral. The results shown in Figure 4.1 are done by multi-scale testing with input dimensions 256 and 512, which is also the testing dimension yielding the highest mAP score on the LISA dataset.

Figure 4.2 shows the detection results on the Multiple Cars Scene dataset, this dataset is quite more challenging than the LISA dataset since some of the license plates are relatively small in the images, making it hard to detect with low input dimension, we used multi-scale testing with dimensions 256, 512 and 1024 for those visualization results since it obtained the best mAP on the Multiple Cars Scene dataset.

Figure 4.1: Classification and pose estimation results of LISA dataset

Figure 4.3 gives some examples of the detection missing issue by vehicle detection-based method mentioned in section 1.3. By utilizing our proposed method, we successfully avoided this issue and found all of the license plates inside the image. This indicates that our method is more reliable when the amount of vehicle inside an image becomes larger since the overlapped situation among vehicles will increase as well.

Figure 4.4 and Figure 4.5 visualize the license plate probability for each pixel and the classification ability of the model by plotting the distribution of those high probability pixels. Our model can locate the license plate within a precise region and tell all the possible pixels for the car's front and rear.

Figure 4.2: Classification and pose estimation results of Multiple Cars Scene dataset

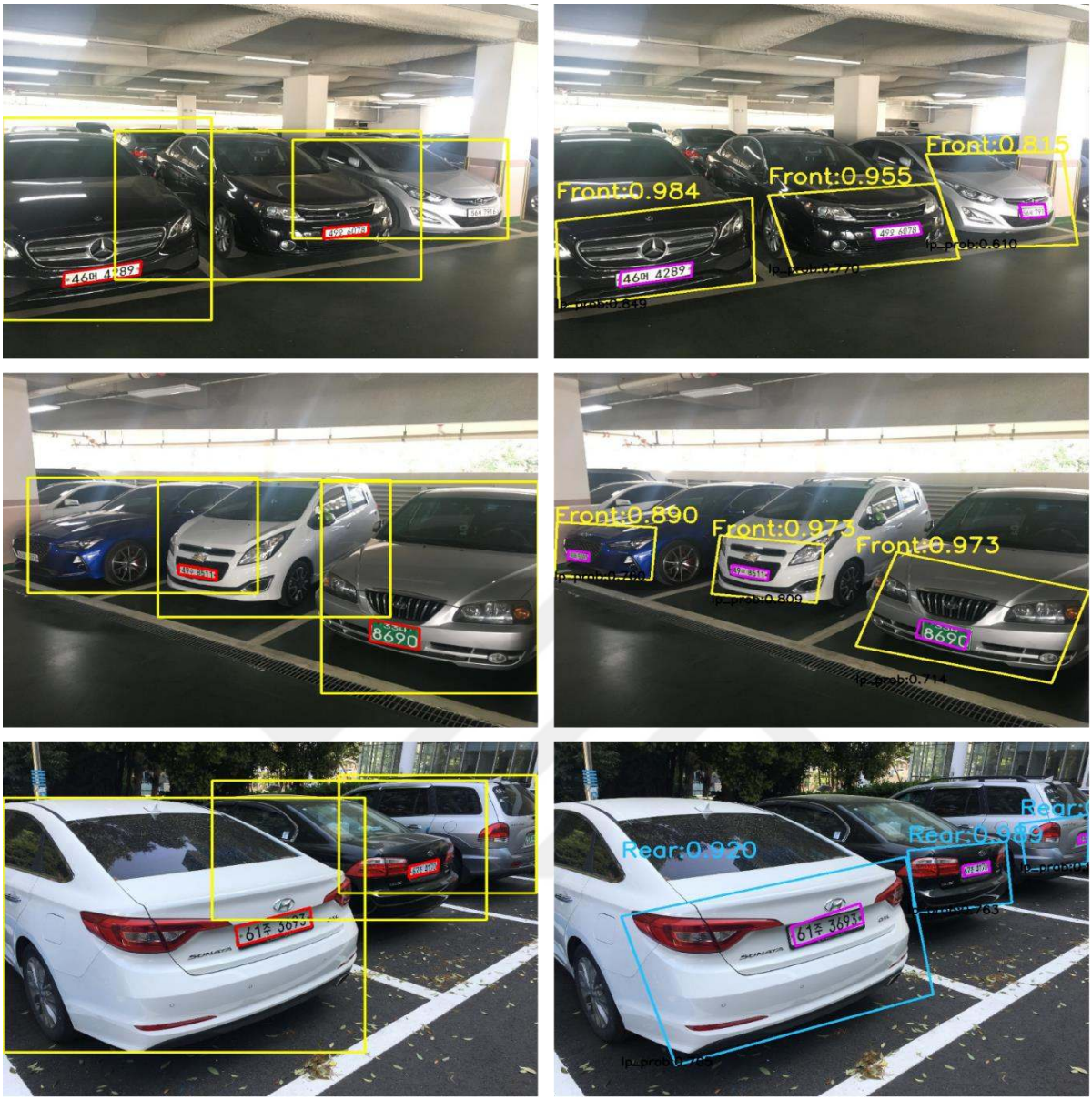


Figure 4.3: Comparison between vehicle pose estimation and license plate localization (left column) and proposed method (right column). On the left side, pose estimation and license plate localization missing appeared due to the false regression of license plate; on the other hand, our proposed method can avoid those cases and find all of the license plates.

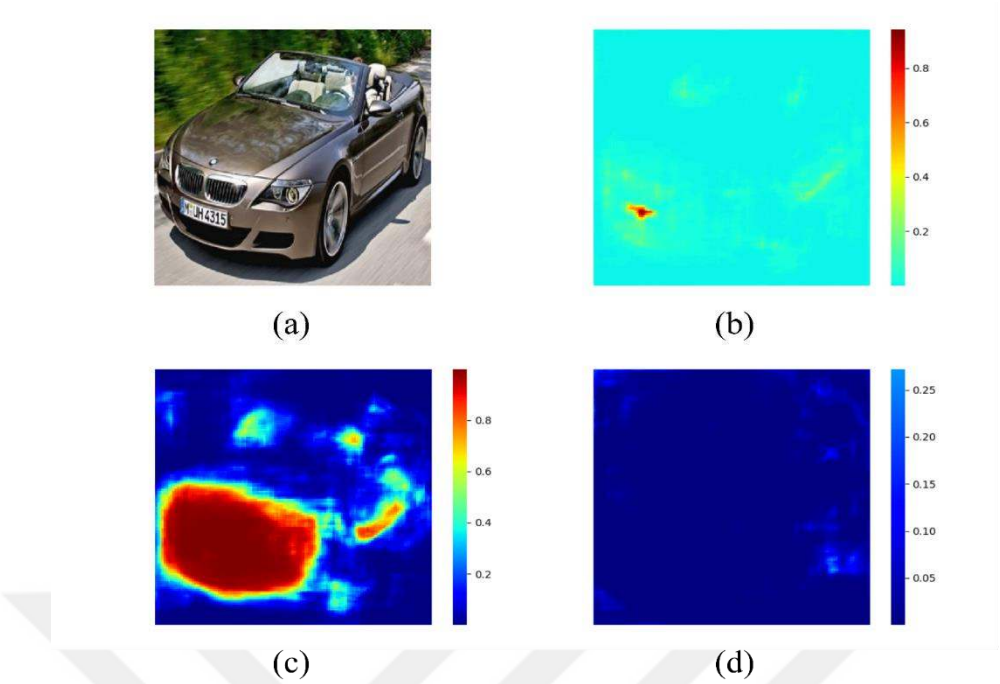


Figure 4.4: Heatmap of LISA dataset. (a): Input image front (b): License plate probability (c): Front probability (d): Rear probability

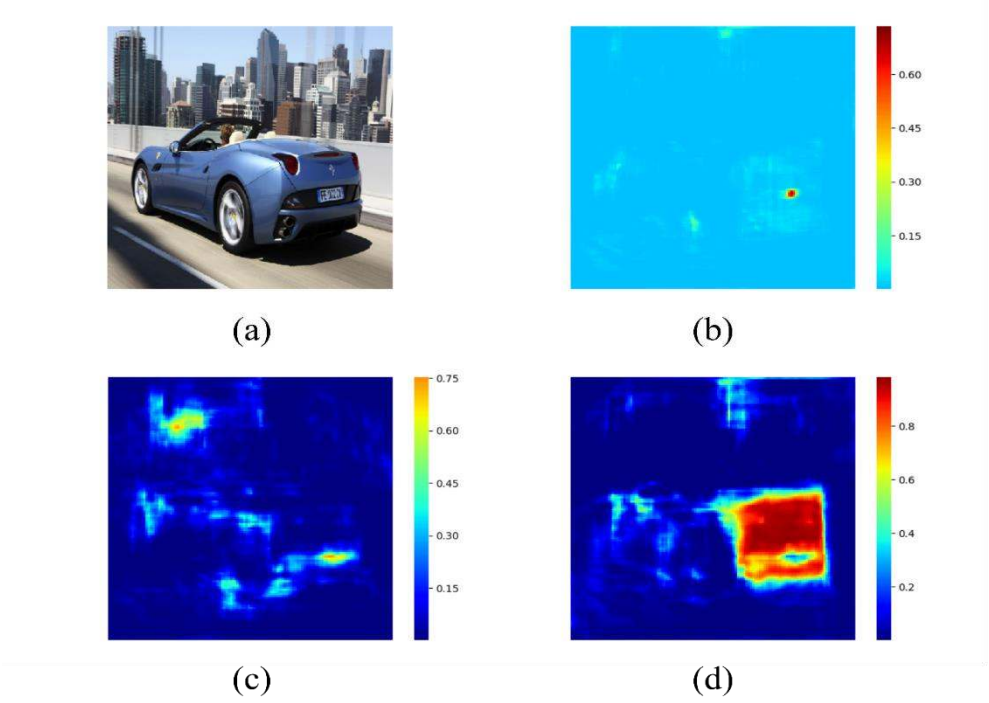


Figure 4.5: Heatmap of LISA dataset. (a): Input image rear (b): License plate probability (c): Front probability (d): Rear probability

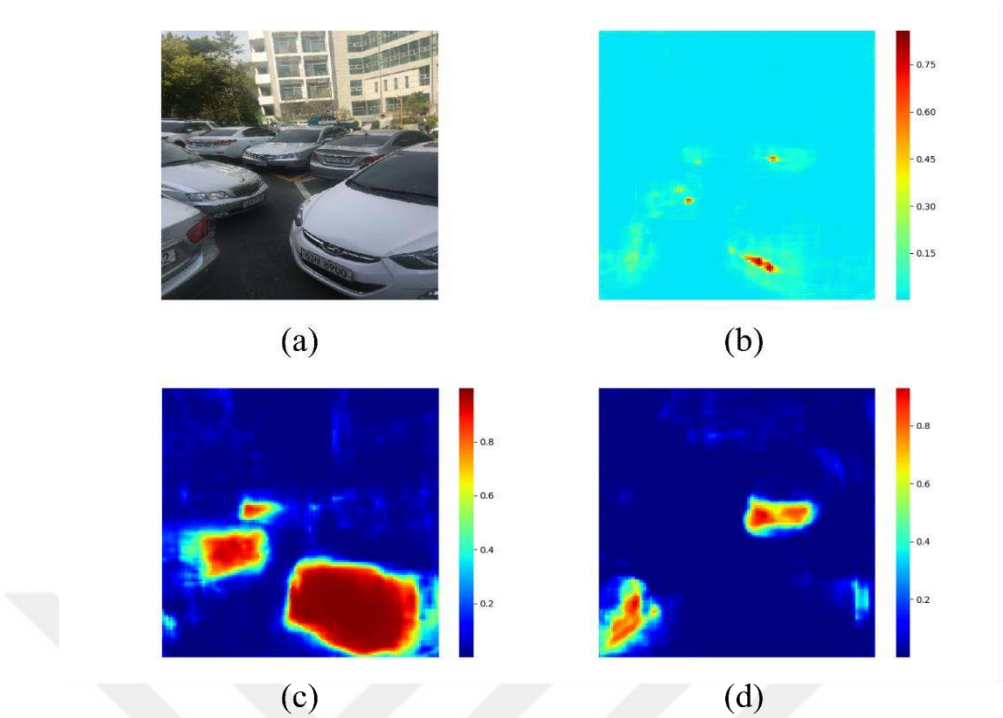


Figure 4.6: Heatmap of Multiple Cars Scene dataset. (a): Input image (b): License plate probability (c): Front probability (d): Rear probability

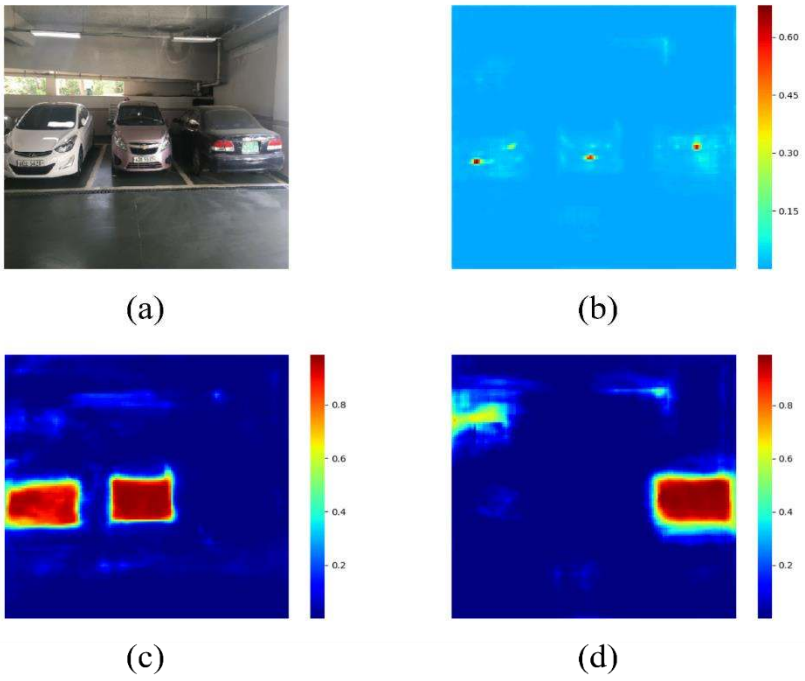


Figure 4.7: Heatmap of Multiple Cars Scene dataset. (a): Input image (b): License plate probability (c): Front probability (d): Rear probability

4.2.2 Quantitative Evaluation

The mAP is often used for the evaluation of a vehicle classifier. The average precision refers to the average value of the precisions with different recalls and is calculated for each class individually. Mean average precision then refers to the arithmetic mean value of the average precisions of all of the classes. To be clear, although we only have one class (license plate) in the model, in our evaluation, we still used the term mAP for consistency. Before moving on to the quantitative evaluation, in the following contents, we will first declare the different mAP metrics that might be confusing.

The early calculation method for mAP is proposed by the Pascal VOC challenge, they calculated the 11 points of precision values by interpolation with recall value from 0 to 1.0 with an interval of 0.1. The average precision is then calculated by the average of those precision values. In Pascal VOC mAP, they treat the detection result as a true positive detection if the IoU with the ground-truth label is larger than 0.5.

Since recent vehicle detection research tended to move to dataset as a benchmark, we used the new metric for mAP calculation proposed. They used 101 points of precision values by interpolation with recall value from 0 to 1.0 with an interval of 0.01. Different from the Pascal VOC metric, they treat a detection as a true positive detection under different IoU threshold settings. The main metric, mAP, refers to the average mAP calculated by 10 IoU thresholds from 0.5 to 0.95 with interval 0.05. mAP50 refers to the mean average precision with IoU threshold = 0.5. While the strictest metric, mAP75, is the mean average precision with IoU threshold = 0.75. We used the calculation method for all of our evaluations; for reference, mAP50 will be the most similar metric compared to the Pascal VOC metric.

Table 4.2 lists the mAP evaluation on the LISA dataset, for the proposed DCNN with a single testing scale dimension 512, we got bounding quadrilateral mAP score better than both commercial systems OpenALPR [49] (system with bounding quadrilaterals) and SightHound [50] (system with bounding rectangles). With multiscale testing, we reached the best mAP score of 40.8, which outperformed OpenALPR by 15.3 mAP. With respect to the bounding rectangle, we also achieved the highest mAP50 score among the systems and a similar level of performance on mAP and mAP75.

Performance evaluations under different camera distances are given in Table 4.3 (near), Table 4.4 (medium) and Table 4.5 (far). We separated the LISA dataset into three sub-datasets of

near, medium and far distance. The camera distance is measured by the ratio between the license plate size and the image size, we used k-means clustering method to separate them with center points 0.077 (near), 0.050 (medium) and 0.035 (far).

Table 4.2: Vehicle pose estimation and license plate localization mAP on LISA Dataset.

Method	Quadrilateral			Rectangle			FPS
	mAP	mAP_{50}	mAP_{75}	mAP	mAP_{50}	mAP_{75}	
OpenALPR commercial	25.5	77.5	11.7	34.9	84.4	17.9	
SightHound commercial	17.3	55.8	2.8	**46.2**	80.7	**43.2**	
DCNN (single scale 256)	28.8	68.6	18.2	31.8	68.7	22.5	28.1
DCNN (single scale 512)	35.4	82.3	29.8	39.3	79.2	38.4	12.9
DCNN (256+512)	**40.8**	**90.1**	**34.1**	45.3	**89.0**	42.4	10.1
DCNN (256+512+1024)	38.6	84.3	31.2	43.2	85.5	37.5	2.7

Table 4.3: Vehicle pose estimation and license plate localization mAP on LISA Dataset (camera distance: near)

Method	Quadrilateral			Rectangle		
	mAP	mAP_{50}	mAP_{75}	mAP	mAP_{50}	mAP_{75}
OpenALPR commercial	**46.5**	**94.1**	39.9	49.9	**94.1**	34.5
SightHound commercial	21.9	57.4	10.5	**48.2**	73.3	**58.2**
DCNN (single scale 256)	44.4	82.9	**40.0**	**50.3**	82.9	54.0
DCNN (single scale 512)	18.9	66.1	9.2	23.0	60.6	15.8
DCNN (256+512)	**41.0**	**88.2**	**32.7**	50.1	**88.2**	54.2
DCNN (256+512+1024)	38.8	85.1	29.8	47.3	85.1	48.6

Table 4.4: Vehicle pose estimation and license plate localization mAP on LISA Dataset (camera distance: medium)

Method	Quadrilateral			Rectangle		
	mAP	**mAP_{50}**	**mAP_{75}**	**mAP**	**mAP_{50}**	**mAP_{75}**
OpenALPR commercial	23.9	74.3	8.9	36.9	87.5	21.7
SightHound commercial	15.4	51.5	2.1	**49.6**	78.2	**53.0**
DCNN (single scale 256)	34.3	82.6	22.4	37.0	81.2	23.1
DCNN (single scale 512)	40.9	85.6	34.9	44.8	86.0	51.5
DCNN (256+512)	**46.1**	**94.7**	**40.5**	**50.0**	**94.7**	51.0
DCNN (256+512+1024)	39.5	85.6	32.8	43.5	90.1	38.6

Table 4.5: Vehicle pose estimation and license plate localization mAP on LISA Dataset (camera distance: far)

Method	Quadrilateral			Rectangle		
	mAP	**mAP_{50}**	**mAP_{75}**	**mAP**	**mAP_{50}**	**mAP_{75}**
OpenALPR commercial	18.4	71.6	5.0	26.1	73.3	8.7
SightHound commercial	19.6	64.5	1.7	**44.6**	**88.1**	31.5
DCNN (single scale 256)	14.3	40.8	3.6	16.0	42.6	9.0
DCNN (single scale 512)	39.4	**86.5**	**38.0**	44.0	80.6	**35.2**
DCNN (256+512)	34.9	82.8	29.8	39.1	79.8	27.7
DCNN (256+512+1024)	**40.1**	82.8	34.3	43.1	79.5	33.0

Table 4.6: Vehicle classification accuracy and Average IoU for Car Pose on LISA Dataset.

Proposed methods	Classification Accuracy	Average IoU
DCNN (single scale 256)	95.8%	**74.5**
DCNN (single scale 512)	**98.8%**	67.1
DCNN (256+512)	97.8%	71.3
DCNN (256+512+1024)	96.8%	67.1

Table 4.6 shows the classification accuracy and average IoU for car pose. The classification results are all higher than 95%, and the best result is 98.8% on the single scale testing with dimension 512. Best average IoU value 74.5 came out at a single scale 256 since the front-rear region of a vehicle can be viewed as a large vehicle, and thus the small testing scale will yield a better result.

Table 4.7: Shows the mAP results on the Multiple Cars Scene dataset. Our method outperformed commercial systems with a large gap. WPOD-NET has the highest mAP/ mAP75 score since it is a two-stage license plate detector and yields more refined bounding boxes. However, for mAP50, our model surpassed WPOD-NET by 14.9 mAP to 84.1 mAP, the critical factor of the performance improvement is that our model avoided the missing license plate issue and push the boundary of recall ability further.

Method	Quadrilateral			Rectangle		
	mAP	**mAP_{50}**	**mAP_{75}**	**mAP**	**mAP_{50}**	**mAP_{75}**
OpenALPR commercial	11.6	50.3	1.1	18.5	57.7	4.4
SightHound commercial	10.7	37.8	3.5	29.5	61.4	21.3
WPOD-NET	**47.5**	69.2	**59.7**	**49.8**	69.2	**64.0**
DCNN (single scale 256)	13.5	37.3	7.0	15.7	40.1	8.7
DCNN (single scale 512)	36.5	70.4	35.2	40.1	72.5	38.5
DCNN (256+512)	37.2	72.2	37.9	41.3	74.3	41.3
DCNN (256+512+1024)	43.3	**84.1**	41.9	48.9	**84.1**	51.7

5. DISCUSSION

The learning states for each functional head are shown in Figure 5.1, Figure 5.2 and Figure 5.3. The states are the results of single-scale testing with a dimension of 512. This section is strongly related to section 3.3.3.

For the vehicle pose estimation and license plate mAP50 performance in Figure 5.1, the model learned fast in the early 300k iterations, after that, the learning state became steady and tended to be overfitting after 600k iterations. The training settings modifications at 330k and 434k described in section 3.3.3 fine-tuned the model slightly to get mAP50 beyond 80, but the other datasets did not significantly benefit the mAP for the LISA dataset since the obliqueness levels in the LISA dataset is not that heavy as the ones in the dataset.

The IoU performance of the front-rear region is shown in Figure 5.2. A leap can be observed after 400k iterations, which was the benefited result from the label encoding strategy at 423k mentioned in section 3.3.3. This gave us an insight that a proper design of the labeling strategy can undoubtedly have a massive effect on the performance of a supervised learning vehicle detector.

The classification task is relatively easy for our model to learn, as shown in Figure 5.3, the classification had already reached high accuracy in the early iterations. Nevertheless, the accuracy is quite low before 156k iterations, that was due to the wrong label strategy in our early design, we labeled the front-rear region by the same method used in region regression, which led to a limited region for ground-true labels. Observing the situation, we modified the label encoding strategy immediately and solved the issue. The strategy has been described in section 3.2.2.

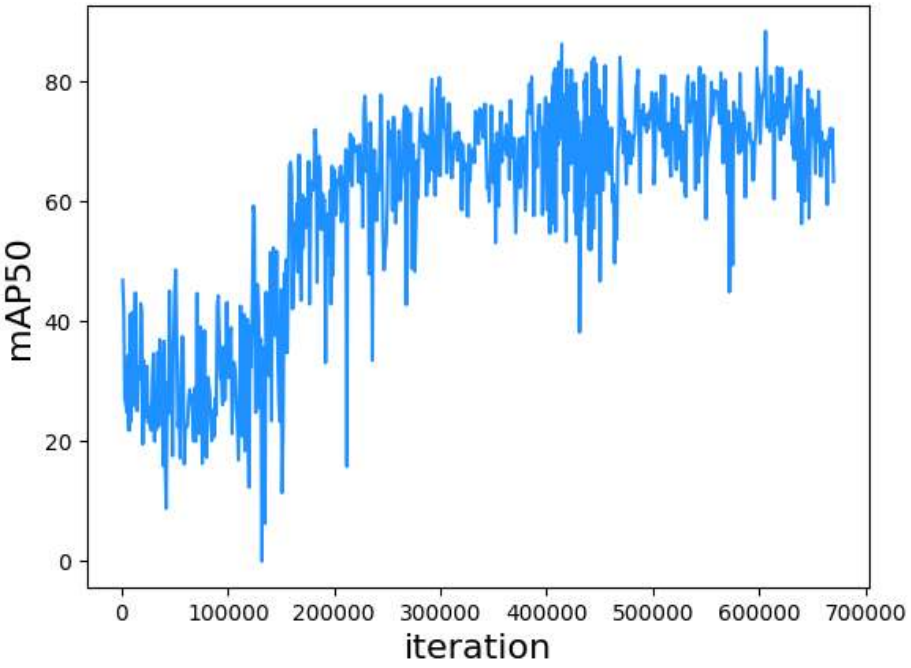


Figure 5.1: Vehicle pose estimation and license plate localization mAP50 to iteration on LISA dataset.

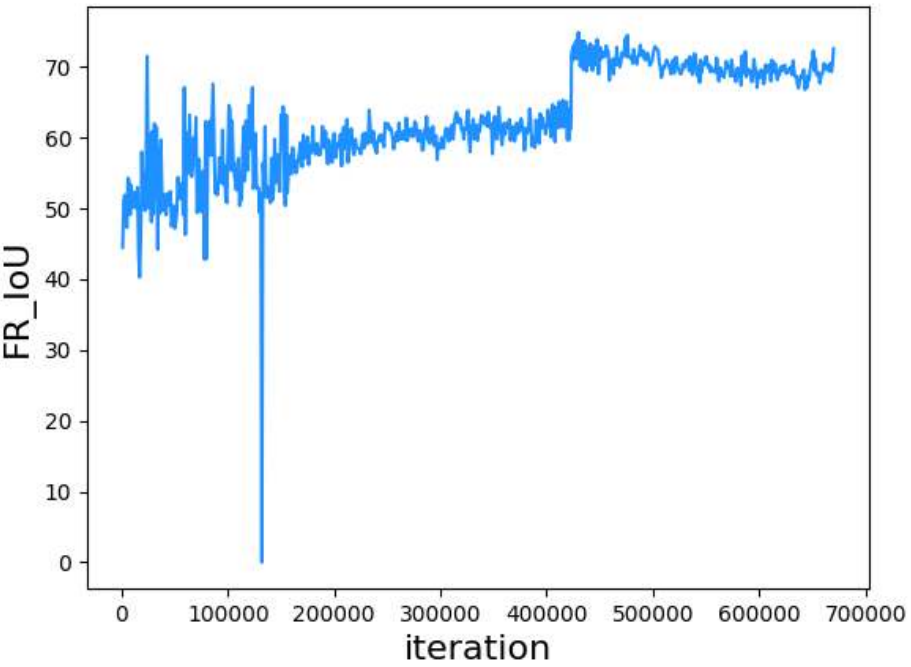


Figure 5.2: Vehicles Front-rear IoU to iteration on LISA dataset.

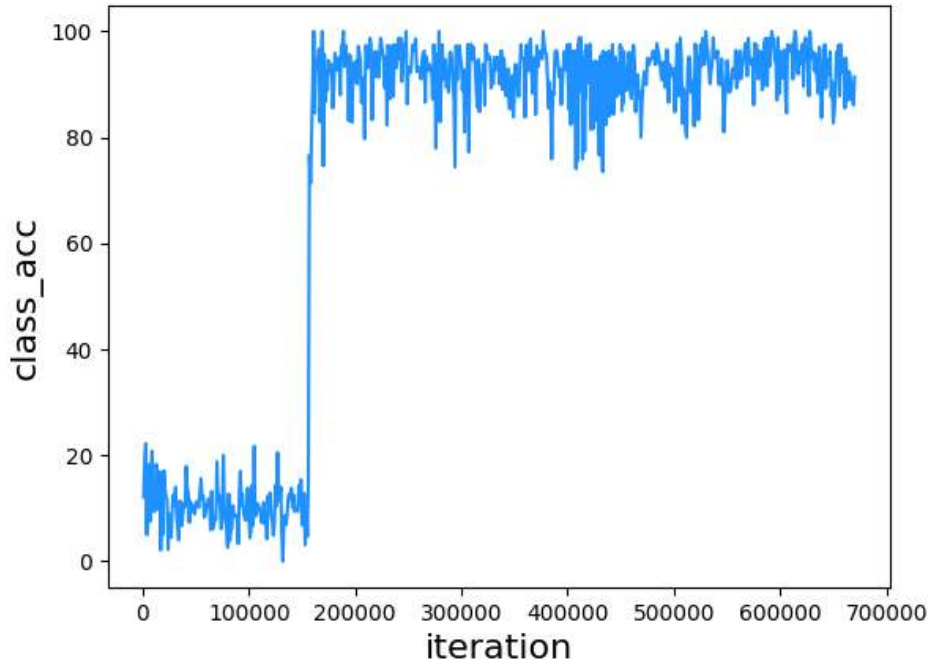


Figure 5.3: Vehicle classification accuracy to iteration on LISA dataset.

Table 5.1: The comparison of accuracy of recognition with existing and proposed techniques

ARTICLE	TECHNIQUE	ACCURACY
[50]	Support Vector Machine (SVM)	80.00%
[51]	Recurrent Neural Network (RNN)	95.70%
[52]	Convolutional Neural Network (CNN)	97.00%
Proposed	Deep Convolutional Neural Network (DCNN)	98.8%

6. CONCLUSION

The proposed vehicle pose estimation with license plate detection model showed the ability to detect license plates under different vision angles. Performance estimation on the dataset with oblique vehicles outperformed the existing commercial systems OpenALPR [52] and SightHound [53], reached mAP/mAP_{50} of 40.8/90.1. Our DCNN based system detects bounding quadrilateral instead of bounding rectangles, yielding a more precise indication for vehicle pose estimation and license plate localization compared to conventional systems.

Another main contribution is providing the vehicle information while performing vehicle pose estimation and license plate localization detection, we called this kind of information contextual information, which provides the relation comprehension between the vehicle pose estimation and license plate localization and the vehicle, we got the pose classification accuracy 98.8% and average IoU 71.3%. Applications like traffic scene analysis, we may utilize contextual information for enhancing the interpretation of the vehicle pose estimation and license plate localization. Since we have obtained the area of the owner car, by further analyzing, we can get; for instance, car brand, model, and color information. In addition, some parking lots tell users to park their cars in a consistent direction (e.g., back-in parking only), the pose information given by our system might help the management of those parking lots.

To accomplish our purpose, we designed a novel anchor-free method for region proposal, which is not limited to the vehicle pose estimation and license plate localization task but can be extended to other vehicle detection tasks. The method might be an inspiration for future vehicle detector design.

The model is designed as a one-stage detector, which is fast enough to perform in real-time, meeting the request for most vehicle pose estimation and license plate localization detection applications. But the drawback exists, as a one-stage detector, finding a highly refined bounding area is more difficult than two-stage detectors do.

A dataset with manually annotated vehicle front-rear regions is used in our work, which is served as a key element to train our model successfully. Dataset is extremely oblique but readable vehicle pose estimation and license plate localizations is used, with the help of the dataset, generalization of applicable scenes for vehicle position estimation and license plate

localization can be further improved. We are looking forward to broader usages of the datasets for applications related to car pose estimation and license plate detection.

For future works, adding more functional heads might be a direction. Our model is flexible since we can add several functional heads after extracting features from the backbone network. Functions like car brand and car model analysis or even more precise car pose estimation given in rotation angle might be trained. Introducing the OCR into the end-to-end network to make a complete ALPR system is also an effort worthy to put.

REFERENCES

[1] A. Shrivastava, A. Gupta and R. Girshick, “Training region-based vehicle detectors with online hard example mining,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[2] J. R. Uijlings, K. E. v. d. Sande, T. Gevers and A. W. Smeulders, “Selective search for vehicle recognition,” International Journal of Computer Vision (IJCV), 2013.

[3] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus and Y. LeCun, “OverFeat: integrated recognition, localization and detection using convolutional networks,” arXiv preprint arXiv:1312.6229, 2013.

[4] S. Du, M. Ibrahim, M. Shehata and W. Badawy, “Automatic License Plate Recognition (ALPR): a state-of-the-art review,” IEEE Transactions on Circuits and Systems for Video Technology 23(2), pp. 311-325, 2013.

[5] Sayanan Sivaraman and Mohan M. Trivedi, "A General Active Learning Framework for On-road Vehicle Recognition and Tracking," IEEE Transactions on Intelligent Transportation Systems, 2010. (pdf)

[6] S. Silva and C. Jung, “Real-time Brazilian license plate detection and recognition using deep convolutional neural networks,” in 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 2017.

[7] M. Everingham, L. Van~Gool, C. K. I. Williams, J. Winn and A. Zisserman, “The {PASCAL} {V}isual {O}bject {C}lasses {C}hallenge 2012 {(VOC2012)} {R}esults,” [Online]. Available: http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[8] R. Girshick, J. Donahue, T. Darrell and J. Malik, “Rich feature hierarchies for accurate vehicle detection and semantic segmentation,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[9] R. Girshick, “Fast R-CNN,” in IEEE International Conference on Computer Vision (ICCV), 2015.

[10] S. Ren, K. He, R. Girshick and J. Sun, “Faster R-CNN: towards real-time vehicle detection with region proposal networks,” in Conference on Neural Information Processing Systems (NIPS), 2015.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu and A. Berg, “SSD: single shot multibox detector,” in The European Conference on Computer Vision (ECCV), 2016.

[12] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi and A. C. Berg, “Dssd: deconvolutional single shot detector,” arXiv preprint arXiv:1701.06659, 2017

[13] J. Redmon and A. Farhadi, “YOLOv3: an incremental improvement,” arXiv preprint arXiv:1804.02767, 2018.

[14] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Doll´ar, “Focal loss for dense vehicle detection,” in IEEE International Conference on Computer Vision (ICCV), 2017.

[15] H. Law and J. Deng, “CornerNet: detecting vehicles as paired keypoints,” in The European Conference on Computer Vision (ECCV), 2018.

[16] C. Zhu, Y. He and M. Savvides, “Feature selective anchor-free module for single-shot vehicle detection,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[17] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang and Q. Tian, “Centernet: keypoint triplets for vehicle detection,” arXiv preprint arXiv:1904.08189, 2019.

[18] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan and C. L. Z. P. Dollár, “Microsoft COCO: Common Vehicles in Context,” in The European Conference on Computer Vision (ECCV), 2014.

[19] Avaialble Online: https://www.forbes.com/sites/louiscolumbus/2018/01/12/10-charts-that-will-change-your-perspective-on-artificial-intelligences-growth/

[20] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in International Conference on Learning Representations (ICLR), 2015.

[21] K. He, X. Zhang, S. Ren and J. Sun, “Deep residual learning for image recognition,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[22] A. Newell, K. Yang and J. Deng, “Stacked hourglass networks for human pose estimation,” in The European Conference on Computer Vision (ECCV), 2016.

[23] O. Ronneberger, P. Fischer and T. Brox, “U-net: convolutional networks for biomedical image segmentation,” in International Conference on Medical image computing and computer-assisted intervention, 2015.

[24] T.-Y. Lin, P. Doll´ar, R. Girshick, K. He, B. Hariharan and S. Belongie, “Feature pyramid networks for vehicle detection,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[25] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[26] S.-H. Kuo, J.-E. Ok and E.-Y. Cha, “Vehicle Front-Rear detection for license plate detection enhancement,” in 15th International Conference on Multimedia Information Technology and Applications(MITA), 2019.

[27] Z. Xu, W. Yang, A. Meng, N. Lu and H. Huang, “Towards end-to-end license plate detection and recognition: a Large dataset and baseline,” in The European Conference on Computer Vision (ECCV), 2018.

[28] Yingfeng Cai, Ze Liu, Xiaoqiang Sun, Long Chen, Hai Wang, Yong Zhang, "Vehicle Detection Based on Deep Dual-Vehicle Deformable Part Models", Journal of Sensors, vol. 2017, Article ID 5627281, 10 pages, 2017. https://doi.org/10.1155/2017/5627281 .

[29] Seyed, M.I.; Hossein, N.M.; Hadi, K.; Yaser, P.F. An Adaptive Forward Collision Warning Framework Design Based on Driver Distraction. IEEE Trans. Intell. Transp. Syst. 2018, 19, 3925–3934.

[30] Vijay, G.; Shashikant, L. Lane Departure Identification for Advanced Driver Assistance. IEEE Trans. Intell. Transp. Syst. 2015, 16, 910–918.

[31] Kim, J.B. Efficient Vanishing Point Detection for Advanced Driver Assistance System. Adv. Sci. Lett. 2017, 23, 4115–4118.

[32] Tzutalin, “LabelImg,” 2015. [Online]. Available: https://github.com/tzutalin/labelImg.

[33] K. Wada, “labelme: image polygonal annotation with python,” 2016. [Online]. Available: https://github.com/wkentaro/labelme.

[34] J. Krause, M. Star, J. Deng and L. Fei-Fei, “3D vehicle representations for fine-grained categorization,” in 4th IEEE Workshop on 3D Representation and Recognition, ICCV, Sydney, Australia, 2013.

[35] Available Online: https://towardsdatascience.com/using-hourglass-networks-to-understand-human-poses-1e40e349fa15

[36] Available Online: https://medium.com/swlh/focal-loss-an-efficient-way-of-handling-class-imbalance-4855ae1db4cb .

[37] Efficient 3D Vehicle Detection using Multiple Pose-Specific Classifiers - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Efficient-localization-and-pose-estimation-in-the-UIUC-database-15-Our-approach-allows_fig1_236667064 [accessed 18 September, 2020]

[38] Available Online: https://www.jeremyjordan.me/object-detection-one-stage/.

[39] Hoang VT., Jo KH. (2019) Modified Stacked Hourglass Networks for Facial Landmarks Detection. In: Nguyen N., Gaol F., Hong TP., Trawiński B. (eds) Intelligent Information and Database Systems. ACIIDS 2019. Lecture Notes in Computer Science, vol 11432. Springer, Cham. https://doi.org/10.1007/978-3-030-14802-7_56

[40] Lan, R., Zou, H., Pang, C. et al. Image denoising via deep residual convolutional neural networks. SIViP 15, 1–8 (2021). https://doi.org/10.1007/s11760-019-01537-x

[41] Available Online: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[42] Available Online: https://blog.datadive.net/prediction-intervals-for-random-forests/

[43] Nguyen, V.N., Jenssen, R. & Roverso, D. LS-Net: fast single-shot line-segment detector. Machine Vision and Applications 32, 12 (2021). https://doi.org/10.1007/s00138-020-01138-6

[44] MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving - Scientific Figure on ResearchGate. Available from:

https://www.researchgate.net/figure/Visualization-of-our-label-encoding-Blue-grid-cells-Red-cells-cells-containing-a-car_fig2_311842448 [accessed 18 Feb, 2021]

[45] Available Online: https://www.mathworks.com/help/driving/ug/train-a-deep-learning-vehicle-detector.html

[46] Online Available: https://missinglink.ai/guides/computer-vision/image-segmentation-deep-learning-methods-applications/

[47] F. Chollet, “Keras,” 2015. [Online]. Available: https://github.com/fchollet/keras.

[48] Rothe, Rasmus & Guillaumin, Matthieu & Van Gool, Luc. (2015). Non-Maximum Suppression for Object Detection by Passing Messages between Windows. LNCS. 9003. 10.1007/978-3-319-16865-4_19.

[49] Available Online: http://cvrr.ucsd.edu/LISA/datasets.html

[50] Abdullah, Saman & Omar, Khairuddin & Sahran, Shahnorbanun & Khalid, Marzuki. (2009). License plate recognition based on Support Vector Machine. 78 - 82. 10.1109/ICEEI.2009.5254811.

[51] Usmankhujaev, Saidrasul & Lee, Seonwoo & Kwon, Jangwoo. (2020). Korean License Plate Recognition System Using Combined Neural Networks. 10.1007/978-3-030-23887-2_2.

[52] Jahan, Nusrat & Islam, Saiful & Foysal, Md. Ferdouse. (2020). Real-Time Vehicle Classification Using CNN. 10.1109/ICCCNT49239.2020.9225623.

[53] Avalilable Online: https://www.sighthound.com/press/sighthound-ai-software-now-reads-license-plates