School of Information Technology and Engineering at the ADA University

School of Engineering and Applied Science at the George Washington University

AUTOMATIC NUMBER PLATE RECOGNITION TECHNOLOGIES FOR AZERBAIJANI VEHICLE REGISTRATION PLATES

A Thesis
Presented to the Graduate Program of Computer Science and Data Analytics
of the School of Information Technology and Engineering
ADA University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Computer Science and Data Analytics
ADA University

By
Kamran Huseynov

April, 2022

THESIS ACCEPTANCE

This Thesis by: Kamran Huseynov
Entitled: *AUTOMATIC NUMBER PLATE RECOGNITION TECHNOLOGIES FOR AZERBAIJANI VEHICLE REGISTRATION PLATES*

has been approved as meeting the requirement for the Degree of Master of Science in Computer Science and Data Analytics of the School of Information Technology and Engineering, ADA University.

Approved:

Dr. Abzatdin Adamov	28.04.2022
(Adviser)	(Date)
Dr. Abzatdin Adamov	28.04.2022
(Program Director)	(Date)
Dr. Sencer Yeralan	28.04.2022
(Dean)	(Date)

ABSTRACT

Vehicle monitoring on road surveillance and parking lot management is one of the global problems in the computer vision field that requires the application of practical and contemporary technologies. The main applications in the computer vision field concerning these problems are called Automatic Number Plate Recognition Systems which are based on ever changing methodologies given the rapid development in Artificial Intelligence and Deep Learning fields. The reason for the broad spectrum of different methodologies is that these systems are in high demand by institutions and private businesses where ANPR systems are frequently upgraded to match the best practices available in order to improve performance, increase speed, reduce cost, and overall maximize the efficiency. Thus, this paper is focused on current practices in Automatic Number Plate Recognition systems and the performance of individual building blocks in the ANPR pipeline. In this paper we will examine in depth the details of contemporary practices in the computer vision field for ANPR solutions, especially, the solutions considering the recent innovations and enhancements in deep neural networks. Even though vehicle monitoring is a global problem, We will specifically focus on vehicle registration numbers issued by Azerbaijan and challenges regarding this selected domain. Given the domain, this paper will finally draw comprehensive comparisons between current ANPR technologies and derive meaningful conclusions based on data, experiments, and results.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Explanation
ANPR	Automatic Number Plate Recognition
ANN	Artificial neural network
AUC	Area under Curve
BoF	Bag of Freebies
BoS	Bag of Specials
CCD	Charge-coupled device
CCTV	Closed-circuit television
COCO	Common Objects in Context
CSP	Cross Stage Partial Network
CNN	Convolutional neural network
DIoU-NMS	Distance Intersection over Union non-max suppression
DPM	Deformable Parts model
FPS	Frames per second
GPU	Graphics Processing Unit
IoU	Intersection over union
mAP	Mean Average Precision
OCR	Optical Character Recognition
PAnet	Path Aggregation network
RELU	Rectified Linear Unit
RGB	Red, Green, Blue
ROI	Region of interest
RPnet	Roadside Parking network
SAT	Self-Adversarial Training
SPM	Spatial Pyramid Pooling
SPP	Spatial Pyramid Matching
TPU	Tensor Processing Unit
VGG	Visual Geometry Group
YOLO	You Only Look Once

1. INTRODUCTION

Traffic control and monitoring is a major global problem. Ever growing road infrastructure leads to installation of an exceeding number of cameras for road surveillance. In today's world it is virtually and practically impossible to manually keep track of the increasing number of traffic cameras and identify all vehicles passing through. Automatic Number Plate Recognition is a straightforward methodology that solves this problem by automatizing vehicle registration identification process which makes it possible for institutions to monitor and govern the roads and also helps investigate specific cases in order to ensure road safety and security. For example, ANPR systems are a core facilitator in automatically identifying and fining drivers exceeding speed limit or breaking traffic laws. ANPR also significantly increases the efficiency and effectiveness of policing institutions of a country by providing the necessary data in order to investigate crimes such as reckless driving, car theft, or human-trafficking. In other words, any kind of automatic solution to govern the traffic and ensure road safety has Automatic Number Plate Recognition as a core element. Now let us further discuss what does ANPR system in reality do for road surveillance and what problem does it address in terms of computer vision.

1.1 Core Definitions

The road surveillance cameras provide a real-time feed of the road, usually capturing the traffic from a higher perspective. When we are talking about real-time feed in general, we can say that having more than 20 frames per second in a video provides a real-time feed since showing 20 images in a second continuously is enough to create a smooth and fluent visual. However, in our case we will assume real-time feed to be 30 frames per second as most of the surveillance cameras provide feed at this speed [1]. So considering what is our unstructured raw data from traffic cameras, in ANPR we consider our input as a single image of the road taken from a surveillance camera, and we do have 20 of these input images for each second. Finally, we would like to emphasize that the main goal of an ANPR system is to extract all vehicle registration numbers present in each image as a string. So, in our problem domain, ANPR system takes an image of a road as the input and returns a list of strings of vehicle registration numbers as the output.

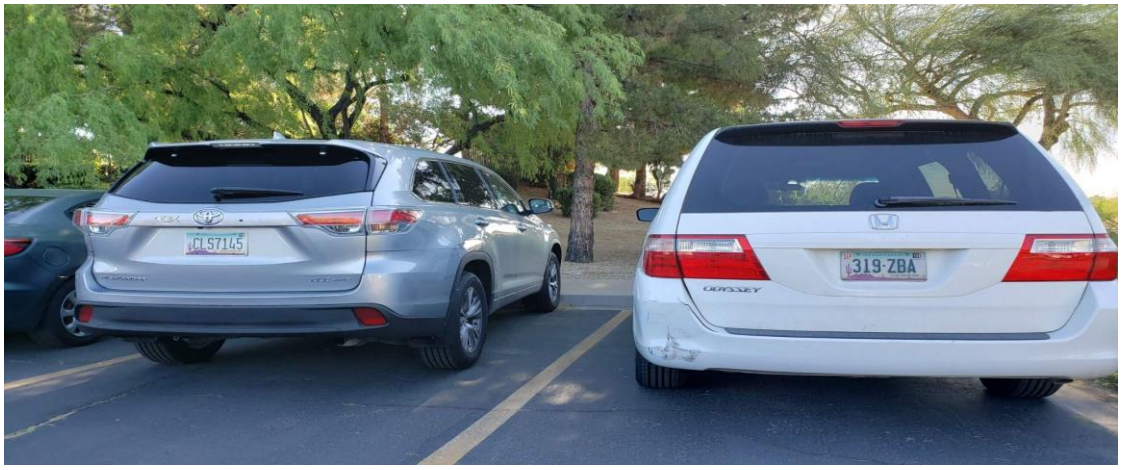


Figure 1: A sample input image for ANPR system

Sometimes it is also possible that ANPR system extracts a few more features from an image, such as the model, type, or color of the vehicle, but the main purpose is to output number plates, and we will assume that ANPR system is solely responsible to determine vehicle registration numbers in this paper for the sake of argument. For example, given the figure 1, we would like to locate two number plates on the input image and use them to extract strings for our output.

Figure 2: Intermediate images of Number Plates for ANPR system based on figure 1

ANPR systems often locate the vehicle registration plates and then derive output from those parts individually. For example, considering figure 1 as our input image, figure 2 shows localized number plates from which the ANPR system finds and returns the final output, which in our case is “CLS7145” and “319.ZBA”.

1.2 Challenges of ANPR

Although Automatic Number Plate Recognition may seem like a straightforward process there are abundant number of challenges associated with ANPR. Many of these challenges are related to the fact that ANPR systems require real world solutions which comes with the obstacles of open environment system for data. Firstly, as we already mentioned ANPR systems are widely used in the monitoring of public roads controlled by government institutions. Having this kind of implementations in ANPR field creates a lot of constraints involving high scrutiny of data sources. Many state-of-art algorithms and neural network architectures are shared openly in computer vision community, however, lack of access to data from real life domains, extremely halts the progress of ANPR systems. Mainly data used in the training of ANPR related models either kept classified by government agencies or privately owned by big tech companies who has business interests contradicting with openly sharing such information. Another problem is that deployment of full scale ANPR system is costly and requires a lot of resources. In fact, for this reason, This paper will be focused on proposal of only minimal viable product versions for ANPR systems. So far, we have mainly talked about logistical issues of

ANPR systems, now it would be a good practice to discuss practical challenges associated with Automatic Number Plate Recognition. Many of these challenges are also discussed in detail by Dahiya et al. in their respective paper which we will also mention [2].

Figure 3: sample input image for ANPR with overloading number of cars

Firstly, even though we usually have few cars present in traffic camera feed, some images that need to be process might contain excessive amount of cars during rush hours with high traffic congestion as shown in figure 3. Such cases might overload the ANPR system requiiring exceeding processing power which can even result in a system crash if not handled carefully. Moreover, there are special cases of noise in the data domain of Automatic Number Plate Recognition which can damage the performance of the system.

(a) (b)

Figure 4: Examples of License Plate Occlusion Problem, (a) Caused by Dust or Dirt, (b) Caused by Gaussian Noise

For example, sometimes it is very challenging to identify license plate number due to occlusion meaning that the characters in license plate are obstacles with foreground noise. Few

scenarios, where License plate occlusion emerges are illustrated in figures 4(a) and 4(b). There are even cases where drivers intentionally keep their license plates as dirty as possible in order to break traffic laws without the fear of receiving penalty or evading toll when using privately owned roads [2].

Another problem with ANPR systems occurs when capturing high-speed roads. When the used camera do not have adequate recording frequency for fast moving cars, this will create footages of phasing out cars where it is impossible to capture or read any vehicle registration plates as demonstrated in figure 5.

Figure 5: . Sample for Problems in ANPR associated with high-speed roads

Therefore, sometimes using traffic monitoring sensors with minimal specifications will be counterproductive, and it is crucial to use traffic cameras with higher refresh rate for the cases such as one shown in figure 5.

Furthermore, there are challenges caused by not the data itself, but the sensor used to gather content for Automatic Number Plate Recognition. Ofter, challenges faced by ANPR system would be the result of using low resolution cameras which are inadequate for the expected task.

Figure 6: Low Quality Imaging Caused by Traffic Monitoring Camera

Figure 6 illustrates 2 scenarios of ineffectively equipped camera such as camira lacking nightvision sensors 5(a) or traffic camera simply having very low resolution.

Another set of challenges are created by environmental or weather conditions. ANPR systems will often struggle to get a clear picture while raining or snowing weather conditions. In windy weathers traffic monitoring cameras often struggle to keep stabilized perspective resulting in excessively blurred images. Harsh weather conditions might also damage the sensors of traffic cameras.

As we have discussed several challenges corresponding to Automatic Number Plate Recognition, now we would like to move on to related work on ANPR field in order to see how different systems have been designed to overcome ANPR specific challenges.

1.3 Related Work

Zhao and Hu try to deal with shortcomings of Haar-like cascade classifier in their original paper [3]. Even though Haar-like cascade classifier has capability to yield high detection rate, the outcomes of such license plate detection algorithm also results in abundance of false positives. To deal with this issue Zhao et al. introduces the use of histogram oriented gradients (HOG) features to determine which of candidadacies proposed by Haar-like cascade classifier are likely to be the actual license plates. The use of histogram of oriented gradients results in massive reduction in false positive outcomes by eliminating registration plate shaped objects and incomplete license numbers. Also, proposed histogram of oriented gradients solution largely reduces the overall training time, being quite robust algorithm at the time. Use of Histogram of Gradients as a logistic classifier on top of Haar-like license plate detector results in substantially improved accuracy score of 0.94 instead of previous performance of 0.66 accuracy score [3]. Employing Histogram of Gradients in ANPR systems for cascading operations proves to be an efficient way of dealing with license plate detection problem.

Kalyanshetti et al. make use of several image processing techniques and machine learning algorithms in combination in order to facilitate smooth ANPR system [4]. The paper especially, introduces use of Maximally Stable External Regions technique for localization of vehicle registration plates. Further Kalyhanshetti et al. demonstrate exceptional work on extensive number of image processing steps to acquire segmented characters from the output of Maximally Stable External Regions methodology. The paper approaches the ANPR problem as general as it is possible with working on license plates having different problems, issued in different countries, having different shapes, sizes, and various ranges of characters from different languages. They introduced their custom Convolutional neural network in order to solve character classification problem. The resulting ANPR design achieves %96 of accuracy.

Chen et al. proposes License Plate Recognition technique consists of two main modules: a license plate locating module and a license number identification module [5]. The former characterized by fuzzy disciplines attempts to extract license plates from an input image, while the latter conceptualized in terms of neural subjects aims to identify the number present in a license plate. The introduced ANPR system manages to get %95.6 accuracy in recognizing license plate numbers of 1027 vehicle registration plates

Kocher and Chevik propose an ANPR system based on artificial neural networks [6]. For their proposed system, authors use 220x50 dimensional images taken from CCD cameras. The

ANPR system employes Canny edge detection algorithm and Binary Large Object coloring algorithm in order to localize vehicle registration plates and detect numbers and letters respectively. Further the digitized characters were classified by using multi layered perceptron neural networks. Two ANN models were used in the proposed ANPR solution where multi layer perceptrons was responsible for classifying numbers and letters separately. The use of two separate ANN models increased the overall performance of the system substantially since it eliminated confusions between characters such as letter “O” and number “0” or letter “B” and number “8” dealing with numbers and letters individually. As a result, the proposed design achieved average of %98 accuracy for its selected domain

XU et al. present a novel network model for ANPR system which can predict the bounding box and recognize the corresponding license plate numbers simultaneously with high speed and accuracy [7]. The authors introduces their novel Roadside Parking net algorithm. Roadside Parking net exploits the Box Regression layer to predict the bounding box for license plate. Then, by referring to the relative position of the bounding box in each feature map, RPnet extracts ROIs from several already generated feature maps, and feed the combined features maps to the subsequent Classifiers in order to perform Optical Character Recognition. The authors also present a large-scale and diverse license plate dataset named CCPD which contains 250 000 unique car images. Their novel Roadside Parking net model provides end-to-end Automatic Number Plate Recognition system which is able to recognize high resolution vehicle registration numbers in 61 frames per second with the astonishing accuracy of %98.5 [7].

2. SYSTEM DESING AND IMPLEMENTATION

Regularly, all Automatic Number Plate Detection systems do have similar characteristics when it comes to overall architecture of the pipeline. It is important to understand that the main technique to solve an ANPR problem is to break down the system to different small processes each having a different and particular set of problems. The reason is. As we mentioned in the introduction part, Automatic Number Plate Recognition requires two main processes, Number plate detection or localization and text recognition. In fact, these two processes can be broken down into smaller steps which we will discuss further in this paper. However, considering these two main processes it is simply not possible to provide an end-to-end machine learning architecture that automatically returns a desirable output.

Figure 7: Sample image of a raw ANPR input

For example, in this given figure 7, if we simply run an end-to-end computer vision algorithm, we do not only get “ALI8 PNE” as an output since we have more text present in the input raw image. In fact, we will get two more texts “ALPINE” and “A1105” as parts of output, and there is no possible way of distinguishing between the text that belongs to the license plate and other irrelevant text visible in the image.

Figure 8: Samples for ANPR images where additional signs and billboards are visible

But this is just a simple case, and in the real world example we will face more such problems since it is not just the text written on cars, and we have to consider overall capture of the traffic camera for the problem. For example, in figure 8 we share other real world scenarios where end-to-end machine learning solutions will struggle again. It is often possible to have billboards or road signs with text captured from the perspective of traffic monitoring cameras where they will confuse our algorithm. Of course, It is possible to add several rules for each camera to exclude such cases. However, considering the global scope of the usage for ANPR systems, maintaining such a system would become a tedious work considering that many systems are expected to cover and process feeds from thousands of cameras.

Therefore, ANPR systems are often broken down into two main building blocks each dealing with different sets of problems.

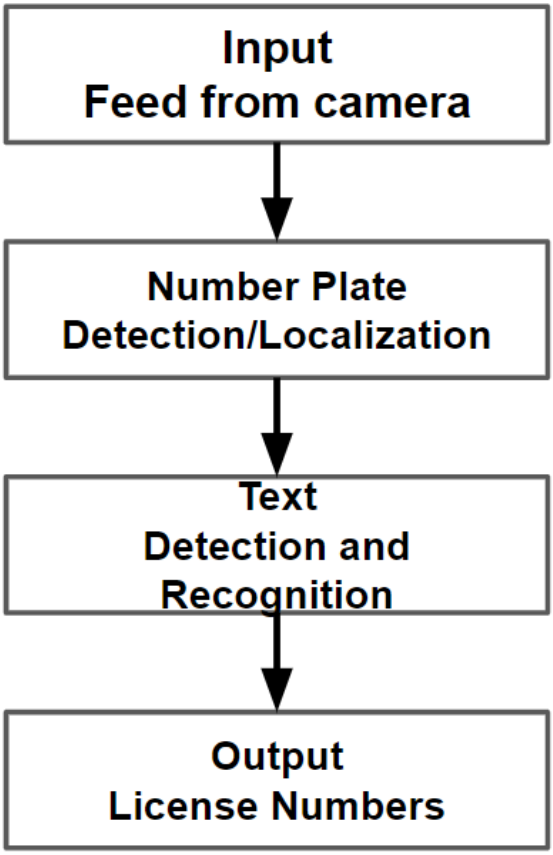


Figure 9: Proposed Pipeline Structure For ANPR System

Figure 5 shows the usual convention for ANPR solution that we are following in this paper. Now we will further discuss and explore different methodologies in this paper employed for each building block, namely, number plate detection and text recognition.

2.1 Number Plate Detection

We will firstly talk about the number plate detection or localization problem which is a process to identify car plate coordinates in the input image and retrieve them for further use. We have already discussed why this is a necessary step in Automatic Number Plate Recognition in the previous section. Now we would like to clarify what method is mainly used in this paper in order to solve this part of the pipeline. Evidently, the number plate detection part of the pipeline is required to have high performance and accuracy. This means that using multi stage object detection models such as R-CNN and Fast R-CNN would make us able to achieve the needed high accuracy rate [8,9,10]. On the other hand, we should also consider real life constraints of an ANPR system. Automatic Number Plate Recognition applications have high speed demand since they are working with real-time feed from traffic cameras. Therefore, we define our static metric of 30 frame per second inference speed for the overall pipeline of an ANPR system. Therefore, using multi-stage object detection algorithms would create a massive bottleneck in the overall architecture that would lead to slow and undesirable

performance. Therefore, for the ANPR system, we considered employing a single step object detection algorithm. In fact, we decided to use one of the most successful object detection models, You Only Look Once. YOLO is a legacy object detection model which comes with different newer versions meeting the demands of advancements in computer vision. For this paper we are specifically using the YOLOv4 model which is known as the last official version of You Only Look Once models.

2.1.1 *YOLO*

Originally, You Only Look Once was presented by Redmon et al. as an end-to-end solution for an object detection problem [11]. Multi-stage object detection models take a lot of time to make an inference on an image since they are using sliding window and region proposal-based techniques, which is restricted to work on each proposed isolated region individually. On the other hand, YOLO is using a unified convolutional neural network which is able to see and infer from an entire input image, thus, implicitly encoding overall contextual information for detecting different classes from an image. Let us first dive into the architecture of the YOLO algorithm to get more insight about the idea of a single Convolution Neural Network as a solution for Object Detection.

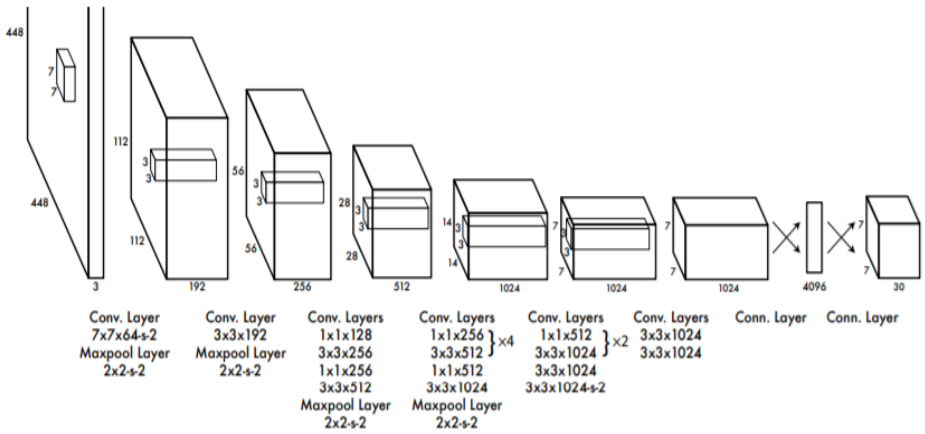


Figure 10: The architecture of YOLOv1 model

Early YOLO model consisted of 24 Convolutional Layers Followed by 2 Fully Connected Layers as illustrated in Figure 10. At that time a much smaller version of this algorithm was referred to as fast YOLO which only had 9 Convolutional Layers instead of 24, and all the other aspects of the architecture were kept the same [11].

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases}$$

Linear activation function is used in the output layer of the first YOLO model, and all the other layers use leaky rectified linear activation function shown in the above equation.

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

And the loss function for early YOLO versions was sum squared error which was fast and easy to optimize. However, this kind of loss function also created problems. Since loss is the sum squared error of each grid cell given in the image and many of the grid cells in each image did not have a class present this would pull down the overall confidence of the prediction. In the given loss function, 1_i^{obj} denotes if an object appears in cell i, and 1_{ij}^{obj} denotes that the jth bounding box predictor in cell i [11]. YOLO solves object detection as a regression problem.

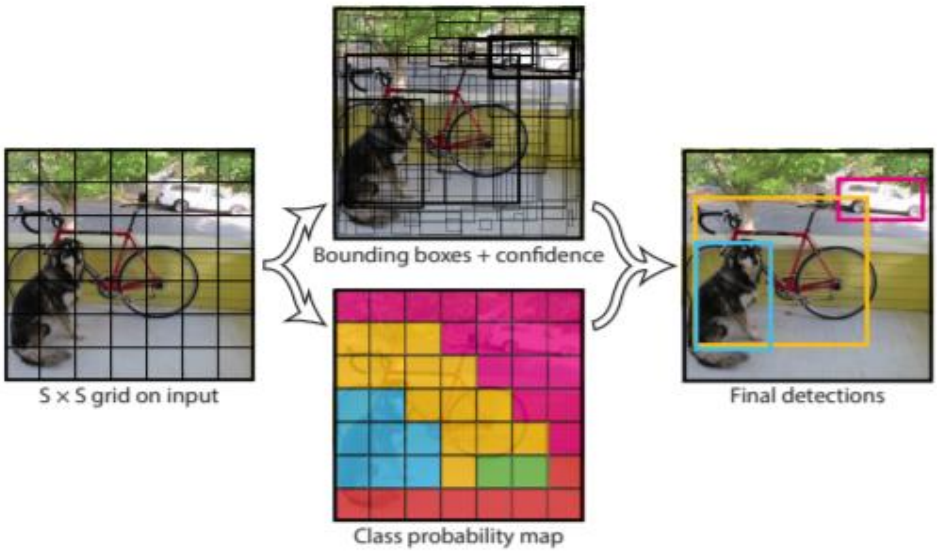


Figure 11: The prediction process in YOLOv1

In the prediction process, the image is divided into an S × S grid where each grid cell predicts B number of bounding boxes, confidence and class probabilities denoted as C. So the prediction layer of YOLO is dynamic and dimensions are calculated as followed:

$$DIM_{output} = S \times S \times (B * 5 + C)$$

So if we had 5 classes for object detection problems where we divided each image to a 5x5 grid and grid cell prediction for the image was 2 bounding boxes, then we would have a 5x5x15 dimensional tensor as an output.

Table 1: Benchmark comparisons of YOLO against other object detectors at the time

Detection Models	mAP	FPS
100Hz DPM	16.01	100
30Hz DPM	26.01	30
Fast YOLO	52.7	**155**
YOLO	**63.4**	45
Fast R-CNN	70.0	0.5
Faster R-CNN VGG-16	73.2	7

As we can also see from Table 1, even at that time YOLO had a decent performance score with much faster inference time. Although Faster R-CNN based on VGG-16 architecture has a very good precision score, it would be useless in our ANPR system since there is no possibility to add it to our ANPR pipeline with 7 frame per second where we at least need to have 30 frames per second for overall prediction time. Note that Redmon et al used a Titan X GPU to drive performance comparisons among these object detection models [11]. Now that we have discussed the overall simplicity and robustness of the YOLO model we will move onto a much more sophisticated and complex version of You Only Look Once that was actually used for our Automatic Number Plate Recognition system, Since YOLOv4 has come a long way from the earliest version, let us quickly recap main advancements till YOLOV4 architecture design.

It is important to note that there have been incremental improvements with each version of YOLO. Main problems of YOLOv1 were localization errors which resulted in inaccurate drawing of coordinates and bounding boxes. The algorithm also struggled to find most of the smaller objects moving in groups in given images such as swarm of bees and flock of birds which resulted in lower recall.

YOLOv2 used several techniques to overcome these shortcomings [13]. One of the techniques used was Batch Normalization which enabled the algorithm to increase average precision more than 2 percent. The technique enabled the algorithm to eliminate all other regularization techniques including dropout that was causing poor performance. Without batch normalization it was not affordable to discard dropout technique since it would cause the trained model to overfit. In YOLOv2 input image resolution was also changed from 224x244 to 448x448 which helped the model to work with higher resolution classifiers. YOLOv2 also employed anchor boxes to get more precise bounding boxes for objects [13].

Later on, the YOLOv3 model achieved incremental improvements over YOLOv2 with introducing new concepts [14]. One of the main advances was the use of independent logistic classifiers for the final prediction layer instead of the softmax algorithm. With this change, now the YOLO model was able to predict two or more classes for the same object which was not possible before. For example, a bus object given in the image could be classified as both bus and

transportation at the same time. YOLOv3 also used another architecture for its backbone which is called Darknet-53. Thich architecture had 53 Convolutional layers instead of 24 layers [14].

2.1.2 *YOLOv4*

Without a doubt YOLOv4 is much more advanced than the earliest versions [12]. It is necessary to consider that while the main author of first three architectures was Joseph Redmon, the main designer of YOLOv4 was Alexey Bochkovskiy. Therefore, the overall design approach in YOLO was changed in version 4. What we mean by that is that the new team working on the YOLO model moved the main overhead of the YOLOv4 design to the training process to increase performance, where testing and inference process was kept fairly simple and robust. This, in turn, allowed YOLOv4 to have very fast inference time with immense performance increase, although the slow training process was the main drawback of YOLOv4.

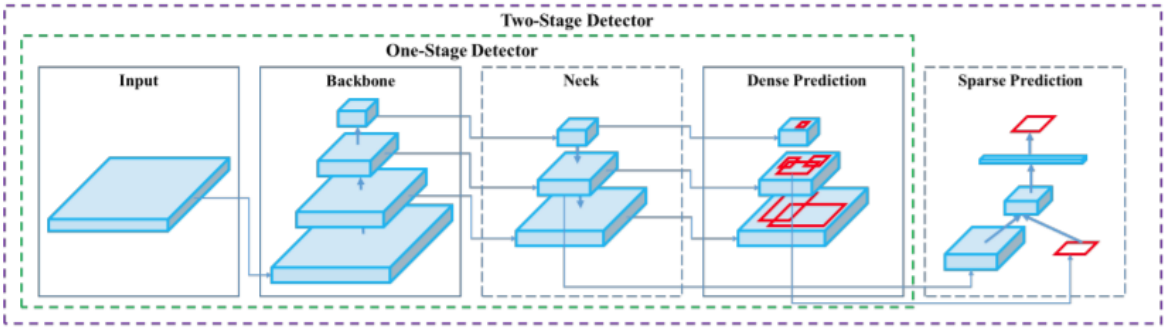


Figure 12: Architecture Overview of YOLOv4 Model

As illustrated in figure 12, the main building blocks of the YOLOv4 object detector were called backbone, neck, and head. Note that the dense prediction section in the shown figure corresponds to the head part of the architecture, and sparse prediction part is only appropriate for multi-stage object detectors whereas YOLOv4 is a single-stage object detection model. In this paper, a vanilla version, the version that was proposed in the original paper, of the YOLOv4 was used. Therefore, in this part we will discuss original choices for backbone, neck, and head building blocks of the model.

Table 2: Architecture of Darknet 53

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Firstly, let us have a look at table 2 where the main structure of Darknet53 architecture is explained. The darknet 53 is a deep learning CNN architecture with 53 layers, including residual skip connections which was used in YOLOv3. Having such residual connections in Darknet53 architecture made YOLOv3 design suffer since such skip connections added a double gradient through signals in the training process [14]. Therefore, YOLOv4 employed CSPDarknet53 architecture in their design that allowed for more smooth gradient flow in the training process and coped with the double gradient problem in this Convolutional Neural Network.

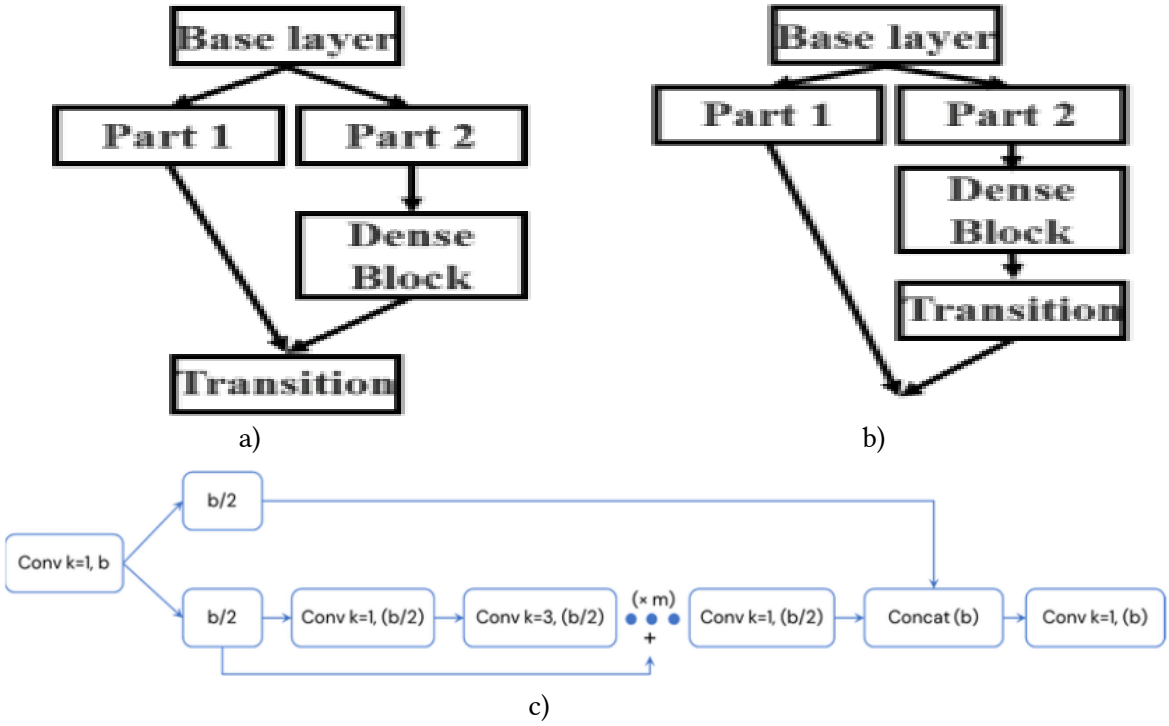


Figure 13: Block of Cross Stage Partial Network. a) Fusion First b) Fusion Last c) Overall Example of CSP strategy

CSPNet stands for Cross Stage Partial Network, which was introduced by Wang et al. [15]. The main idea of CSP strategy is to split the base layer feature map into two partitions which are then merged through a cross-stage process with the technique illustrated in figure 13. So combining the CSP block with Darknet53 architecture gives us the CSPDarknet53 network which was originally used in YOLOv4 design. CSPDarknet53, the backbone of YOLOv4, is responsible for extracting deep features from input images by downsampling the image.

The neck of the YOLOv4 model consists of two parts, namely, additional blocks and path aggregation network. Additional blocks in YOLOv4 are used on top of the backbone in order to enhance the receptive field of the feature map. For this part Spatial Pyramid Pooling which is a convolutional neural network adoption of Spatial Pyramid Matching introduced by Lazebnik et al. where SPP uses pooling layers instead of bag of words method that was used in SPM architecture [16].

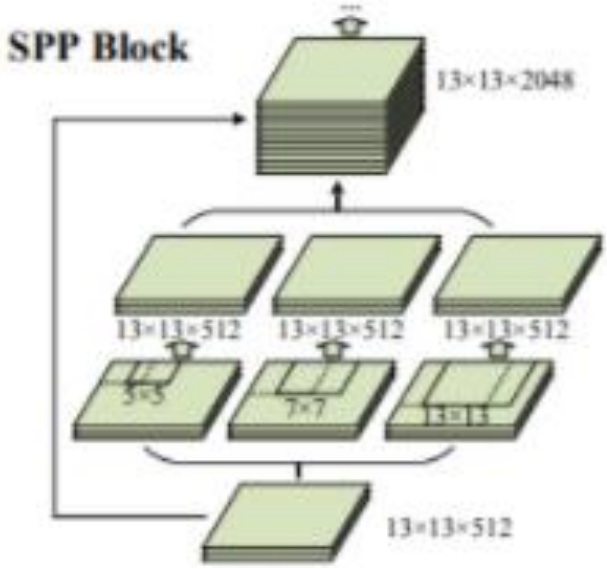


Figure 14: Custom Spatial Pyramid Pooling architecture used in YOLOv4

However, simply using SPP would result in a fully connected layer which was not compatible with the overall design of YOLOv4. Therefore, custom Spatial Pyramid Matching employed in YOLOv4 makes use of concatenation of max pooling outputs with the kernel sizes of 1,5,7, and 13 as illustrated in figure 14. Addition of SPP network to the overall model design resulted in massive performance boost with a small overhead of complexity which was 0.5 percent of the overall inference speed.

The main part of the neck structure in YOLOv4 is the feature aggregation network. The responsibility of the feature aggregation network is to collect, process, and prepare feature maps from the different stages of the backbone architecture in order to feed them into different stages of the head/detector architecture. Vanilla YOLOV4 is a modified version of Path Aggregation Network proposed by Liu et al. [17].

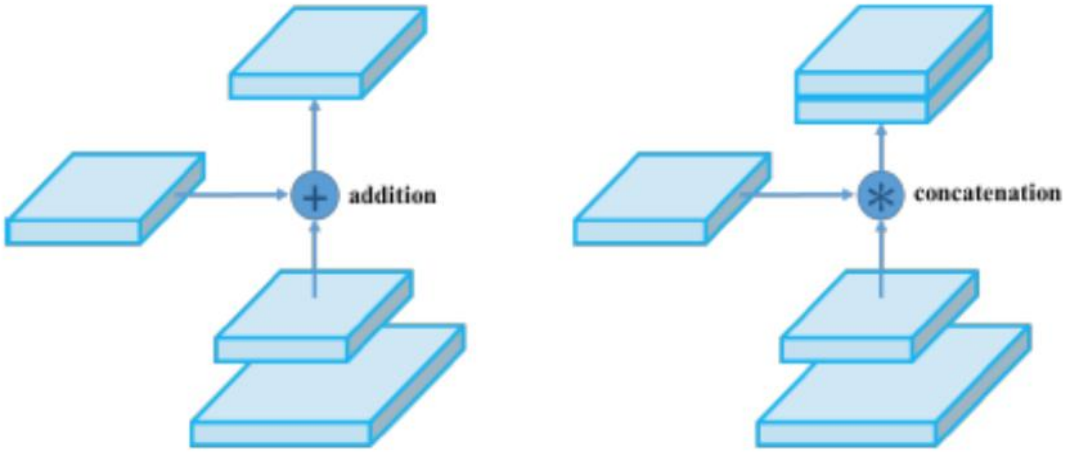


Figure 15: Path Aggregation Network Original and Modified Version for YOLOv4 Respectively

Figure 15 illustrates the key difference of the custom Path aggregation Network used in the neck of the YOLOv4 model. Path Aggregation Network consists of Bottom-up Path Augmentation, Adaptive Feature Pooling, and Fully-Connected Fusion which are shown in the figure 10 in a bottom-to-top manner. The main difference is that customized PANet in YOLOv4 uses concatenation operation instead of addition in Adaptive Feature Pooling process.

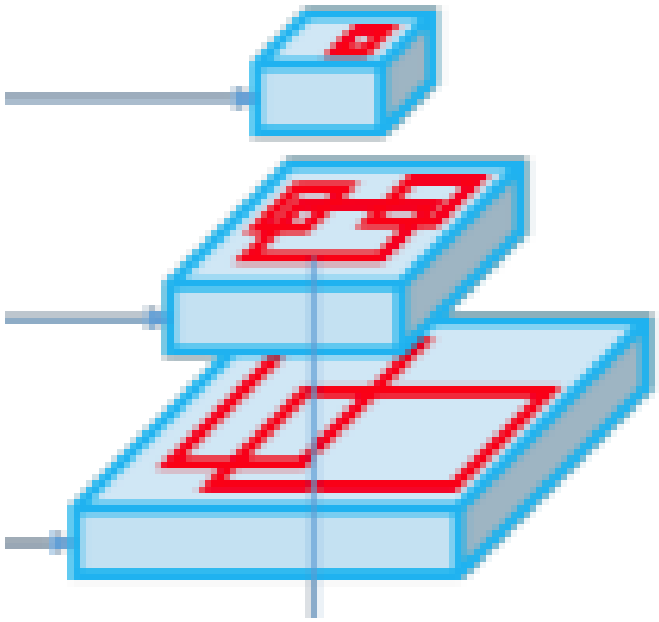


Figure 16: Head of the YOLOv4 as dense predictor

Figure 16 illustrates the head structure of YOLOv4 architecture which is responsible for locating and classifying bounding boxes for object detection. In fact, the YOLOv4 head is the same one with YOLOv2 and YOLOv3 models.

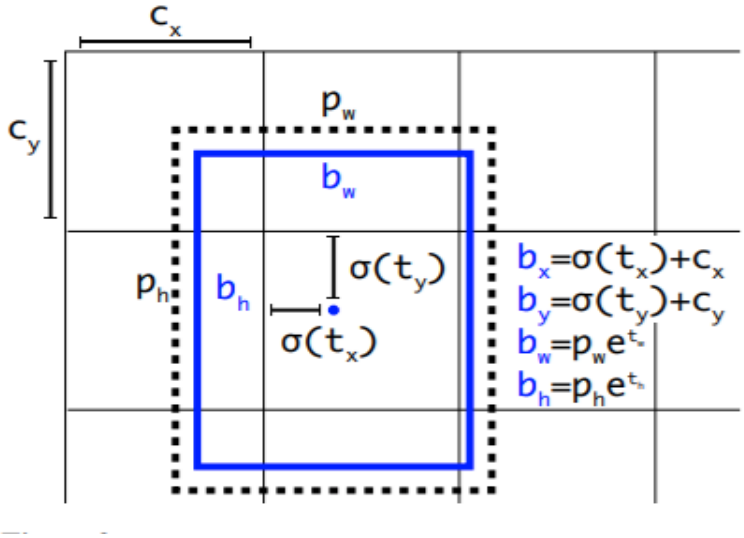


Figure 17: Detection and Classification Process in the Head Structure of the YOLOv4

As we mentioned before YOLO is a single-stage object detection model, thus, the head of the YOLOv4 model locates and classifies bounding boxes on the fly using anchor boxes. Width and height of the boxes are calculated as offsets from box centroids, $\boldsymbol{b_x}$ and $\boldsymbol{b_y}$ as shown in figure 17. On the other hand, center coordinates, $\boldsymbol{b_x}$ and $\boldsymbol{b_y}$, are calculated relative to the location of the filter application based on sigmoid function [12]. The head of YOLOv4 is the part that returns our output layer which consists of isolated outputs for each bounding, tx,ty,tw,th, and cc where cc stands for the confidence level of the object belonging to each class.

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h}
\end{aligned}
$$

Since YOLOv4 gives coordinate outputs with factors proportional to image size, it needs further conversion to get the true coordinates of bounding boxes. Image dimension size for width and height are given as 1 in YOLO and tx,ty,tw,th can range from 0 to 1, This actually creates the ability to feed images with any dimensions to the model, instead of resizing all images to fixed size. Here bx,by,bw,bh refers to bounding box centroids and width-height from centroid to 4 sides of the bounding box. In the equation given above, cx and cy stands for offset to the centroid from top-left corner of the input image, and pw and ph original width and height of the bounding box. So two wrap up the architecture of YOLOv4 used in this paper the order of the building blocks is as follows.

INPUT → CSPDARKNET53 → SPP → PANet → Anchor Based Detector → OUTPUT

As we discussed before, the approach of the YOLOv4 to the overall solution was different than earlier versions since it emphasized on less overhead in inference and gaining high performance based on training process. Indeed, Bochkovskiy et al. introduced two new state-of-the-art concepts that makes YOLOv4 much more elegant, bag of specials and bag of freebies [12]. Now we would like to further discuss these two concepts and emphasize their importance to our work in this ANPR system. Bag of specials are the additions made to our model which provides performance boost to object detection predictions by adding marginal overhead to the inference time. In this case, this marginal overhead to the speed of the model is accepted given that performance increase substantially outweighs the drawbacks in speed. We have actually already discussed some key Bag of Specials additions to our model. Cross Stage Partial Networks, Residual Connections, Spatial Pyramid Pooling, and Path Aggregation Model are all considered Bag of Specials. There are 2 more BoS used in the paper we would like to further discuss. Firstly, the YOLOv4 model uses a much more complex activation function for Convolutional Layers.

$$f(x) = x \tanh(\mathrm{softplus}(x))$$

$$\text{where } | \ \mathrm{softplus}(x) = \ln(1+e^{x})$$

Equation given above is the activation function used in YOLOv4 CNN layers which is called Mish activation function [18]. Through trial and error, the original paper found that the

YOLOv4 model performs best with this activation function as opposed to leaky relu which was used by older versions of YOLO.

The final Bag of Special we would like to note is the use of Distance IoU non-max suppression instead of standard IoU-nms.-maximum suppression. IoU loss is usually given as the below expression

$$\mathcal{L} = 1 - IoU + \mathcal{R}(B, B^{gt})$$

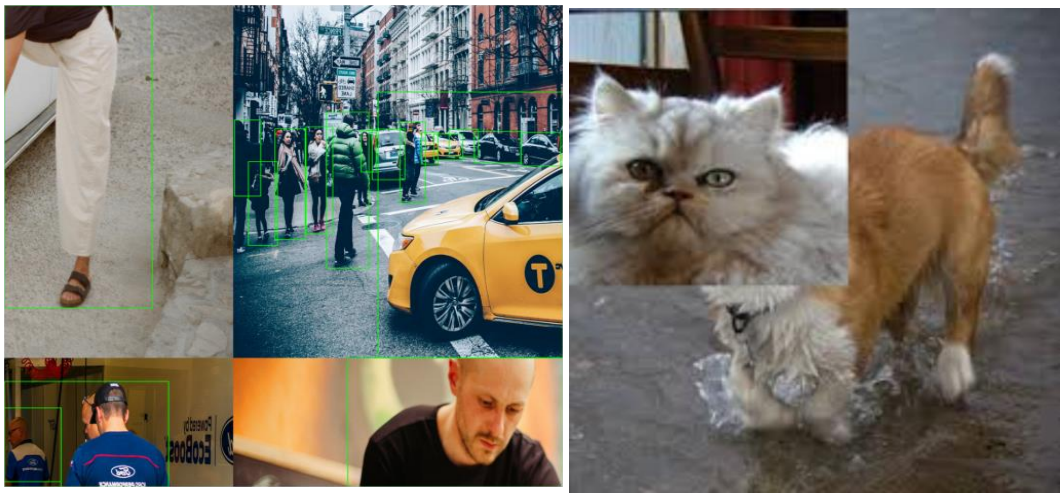
Where $R(B,B^{gt})$ is the penalty term for the predicted box. In normal IoU non-max suppression the penalty only considers the size of the bounding box.

However, in DIoU-NMS we defined penalty as

$$\mathcal{R}_{DIoU} = \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2}$$

where b and b^{gt} are the central points of bounding boxes B and B^{gt}, p() is the euclidean distance and c denotes the diagonal for the smallest box enclosing both B and B^{gt} [19]. This type of penalty in non-max suppression helps us to select the smallest box enclosing the object and which is centered with the centroid of the object. Therefore, YOLOv4 can provide very good bounding boxes for objects with minimal IoU loss.

On the other hand, a bag of freebies does not create any overhead on inference speed of the YOLOv4 model. These concepts are simply designed for training processes only and mainly focused on helping YOLOv4 have a better training process in order to increase accuracy.

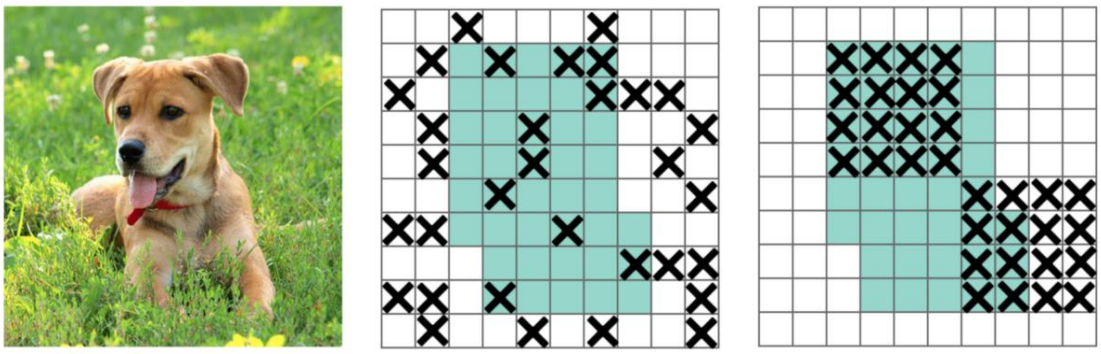


(a) (b)

Figure 18: Bag of Freebies: Data Augmentation Methods (a) Mosaic and (b) CutMix

Two of the BoFs used in YOLOv4 are CutMix and Mosaic which are largely related to data augmentation. Figure 18 shows samples for Mosaic and CutMix data augmentations. As illustrated, CutMix data augmentation is the combination of two input images with different object classes given as one input image whereas Mosaic data augmentation combines 4 images

with different objects together. These kinds of augmentations help YOLOv4 identify out of context objects and learn main features for each object.



(a) (b) (c)

Figure 19: Regularization Methods: (a) Input image, (b) Dropout Method (c) Drop Block Method

Another BoF used in YOLOv4 is the Drop Block regularization method. We are already familiar with the standard dropout method where random pixels from a feature map are ignored for each image. However, YOLOv4 uses Drop Block method where connected chunks of pixels are dropped from each image which actually performs way better than Dropout method. Due to such regularization the model learns to find objects with lost information such as a man standing behind a door or half of the car blocked by a bigger truck. In fact, YOLOv4 is one of the best models in detecting background objects thanks to the Drop Block regularization. We believe that it will also help our object detector to localize license plates with ease in case of minor obstructions.

Another BoF employed in YOLOv4 design is SAT, Self Adversarial Training. Self Adversarial Training is much similar to Self Adversarial Learning which is implemented with 2 forward-backward stages. However, instead of altering main weights learned by model in the first stage, SAT alters the original image excluding parts that were the most important features in detecting the object. This kind of deception helps the model to learn the object better with altered images in the next stage. Note that this is not a whole new architecture but just an augmentation trick to force the model perform better that does not create overhead for the inference speed of YOLOv4.

Figure 20: Bag of Freebies concepts used in YOLOv4

Other BoF elements used in YOLOv4 are but not limited to class label smoothing, Using multiple anchors for the same ground truth, optimal hyperparameters, and random shapes. Some of these elements are illustrated in figure 20. Now that we have covered the overall design of the number plate detector of our pipeline, let us move to the Text Recognition part of our ANPR system.

2.2 Optical Character Recognition

As we have already discussed before, this part of the pipeline in our ANPR system is concerned with locating text or characters in an image extracted by a number plate detector and classifying these characters. For this part of the system we actually created two different versions which might be helpful to achieve better accuracy.

2.2.1 *Using Object Detection Model for OCR*

First of them is our novel way of approaching the problem as an object detection problem rather than Optical character recognition. In this version of the pipeline we are considering using another YOLOv4 object detection trained on 35 classes, 25 characters of the Latin alphabet excluding "O" and 10 number characters from 0 to 9. In fact, intuitively, if we use object detection to detect all present characters in Azerbaijani license plates we can solve the overall OCR problem with an end-to-end solution.

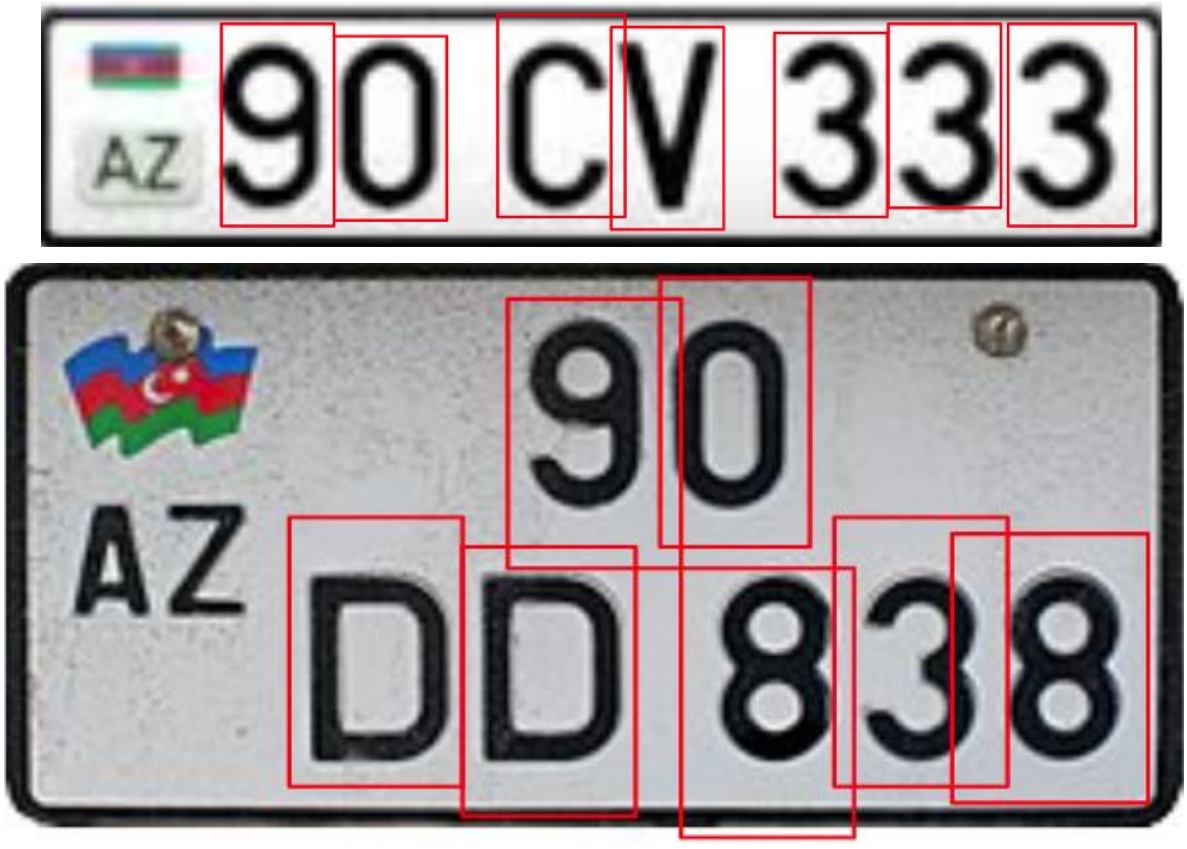


Figure 21: Samples for Solving OCR Problem with Object Detection Model such as YOLOv4

For example, given the output in figure 21 we can use the centroids of classified objects in the license plate to sort the characters and get the required contents of the vehicle registration numbers. Since we have both one-line and two-line text in our license plates based on their shapes, rectangle or quadratic, we would use both c_x and c_y coordinates of our outputs. my centroid needs to have an additional 5 pixels offset since vertical centroid points of objects may vary depending on the position of the found local bounding boxes. For example, in the 2nd image in figure 16, in two parts "09" and "83" we have the latter character with slightly higher vertical centroid point than the bounding box of the former character. Therefore, without additional marginal offset defined we would end up with "09DD388" as our final result instead of "90DD838". Considering this approach we could get very good results by using another YOLOv4 object detection model trained solely on license plates in order to perform Optical Character Recognition.

2.2.2 *Character Segmentation*

Our second way of approaching this part of the ANPR pipeline requires us to divide this stage of the architecture into smaller sets of solutions in order to achieve robust and highly accurate algorithms. In this version we would need to pre process our image in order to locate each character within the image and use a character classification model on segmented characters. In this case segmentation part of the work requires additional processing in order to get good results. Therefore, we will first discuss character segmentation and each step involved in this process in detail.

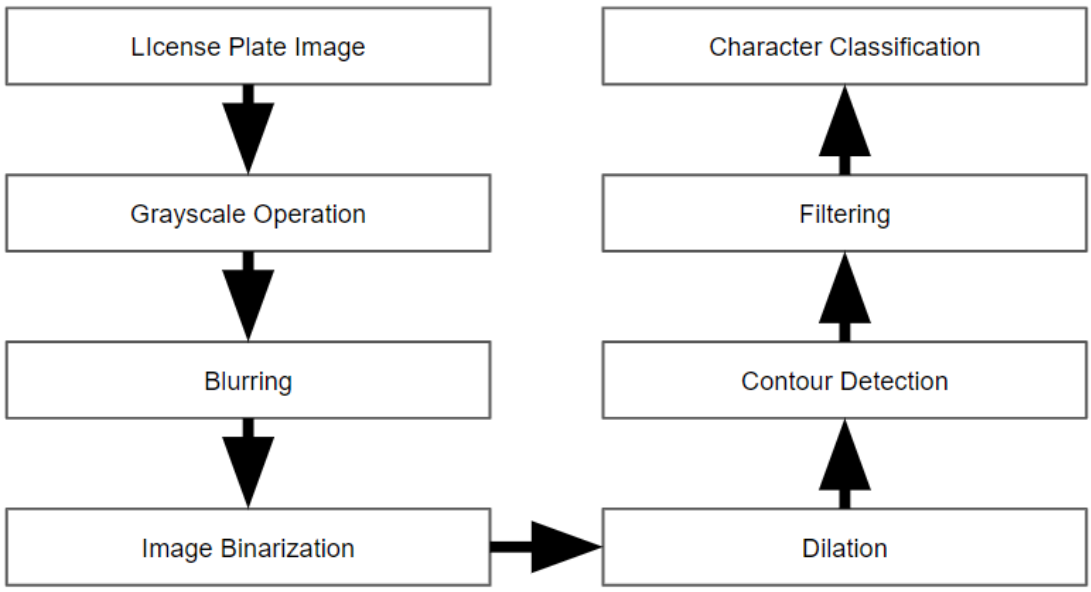


Figure 22: Processing Steps of Character Segmentation and Classification

Since our license plate image detected by object detection algorithm comes in RGB values, we need to first convert this image into one channel grayscale image. The operation to do this

is quite simple and straightforward where we multiple all red, green, and blue pixel values from image by their respective factors and aggregate them into one grayscale channel [23].

$$C_{x,y} = \sum_x^W \sum_y^H 0.3 \times R_{x,y} + 0.59 \times G_{x,y} + 0.11 \times B_{x,y}$$

Here, C refers to our new grayscale channel whereas R,G,B refers to respectively red, green, and blue channels from the original image. W,H are the input image's width and height, and x,y refers to each pixel location in our image. We will now move on with showing these operations in the with the two templates illustrated in figure 16, one for each quadratic and rectangular cases of license plates.

(a) (b)

Figure 23: Example of Grayscale Operation (a) Original License Plates (b) Results After Grayscale Operation

Figure 23 shows the results of our grayscale operation. For the second process, we need to blur the license plates. The reason is to smooth out the edges of characters and also prevent aliasing of the image [22]. Therefore, we have decided to use Gaussian Filtering in order two achieve our objective with blurring. Gaussian blurring updates the pixel values based on the Gaussian distribution using the weighted average of surrounding pixels.

Figure 24: Example of Gaussian Filtering Operation

In figure 24, we can see the results of applying Gaussian blur to our grayscale license plates. When we compare the blurred versions of license plates to their grayscale state, we can easily see that characters now seems to have much smoother edges. In real life camera feed we might also have dirt or dust on vehicle registration plates, using Gaussian filtering also helps to remove noise from images in such cases.

Since we are preparing our license plate images for segmentation now we can move onto image binarization. We would like to have only 0's and 1's in our pixel values as opposed to unsigned 8-bit integer values. Since we only care about locating characters in our image, all character pixels are 1 and all other parts of plate has 0 pixel value. We are using simple thresholding for binarization process, where all

$$p(x,y) = \begin{cases} 1, & if\ p(x,y) \geq T \\ 0 & otherwise \end{cases}$$

Here, p(x,y) stands for pixel value of the image in x,y coordinates and we are using threshold T=128 for our character segmentation.

Figure 25: Example of Threshold Binarization Operation

In figure 25, we can see the results of our binary operation with threshold value of 128. Before contour detection in our image we will use image morphology operation, dilation, on our licen plate where we will be extending the the values of all pixels with value 1 to their surroinding pixels in order to segment license plate characters better by expanding the boundaries of characters. Also, after binarization, some pixels inside characters might lose their values and create a pepper like noise. As we can see in figure 20, character "Z" in the bottom image clearly shows presence of such a noise.

$$\delta_B(S) = S \oplus B = \cup_{b \in B} S_b$$

The dilation operation is shown as $\oplus$ in above equation where S is the original image and B corresponds to structuring element/kernel used [24]. Dilation operation also can be described in terms of Union.

Figure 26: Example of Threshold Binarization Operation

After the dilation operation we remove pepperlike noise from our characters and obtain character with expanded boundaries as illustrated in figure 26. After the dilation operation the license plates are completely pre-processed in order to further apply character segmentation. Since we already have our binary image, we first apply contour detection. Further, we need to filter found contours by using the general attributes of our characters. We define two attributes that help us to filter the contours found in the image. First one is the width-height ratio of the characters.

$$C(w,h) = \begin{cases} C, & \text{if } 1.5 \leq w/h \leq 3.5 \\ 0 & \text{otherwise} \end{cases}$$

C refers to the found contour where as w,h are respectively width and height of the contour. We also filter the remaining contour by their ration of size to the real image size.

$$C(w,h) = \begin{cases} C, & \text{if } w \times h \geq 0.1 \times S \\ 0 & \text{otherwise} \end{cases}$$

We will only allow contours which have size, $w \times h$, larger than %10 of the image size, S. The filterin process pretty much finalizes our cahracter segmentation process. This kind of pre-processing and character segmentation steps might seem a tedious job, however, this is very

robust and high speed algorithm since we do not have many calculations to locate characters. Also, it is possible to use this process asynchronously in order to process batch of license plates at the same time in our ANPR system.

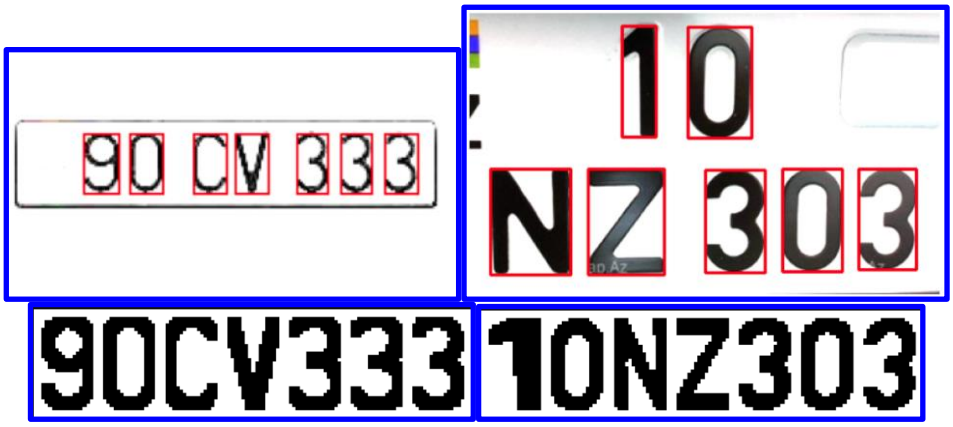


Figure 27: Final Output of Character Segmentation

Final output of the character segmentation illustrated in figure 27 enables us to move on to final move to the character classification part. Now we will discuss in detail how we implemented character classification for our Automatic Number Plate Recognition.

2.2.3 *Character Classification*

For ANPR system we are using Convolutional Neural Network in order to classify segmented characters. We will specifically use MobeliNetV2 model for this classification problem as it is lightweight and can produce pleasent results for our Automatic Number Plate Recognition. MobileNetV2 architecture is based on thin bottleneck layers in which shortcut connections are constructed from inverted residual structures [20]. The reason MobileNetV2 model is so robust is that convolutional layers in this architecture differ from conventional CNN that we use in mose popular networks. First of all, MobeliNetV2 architecture makes use of depth-wise separable convolutions. depth-wise separable operations was first introduced to CNN architecture by Francois Chollet which revolutionized Inception models due to robustness and high-speed it provides to the architecture [21].

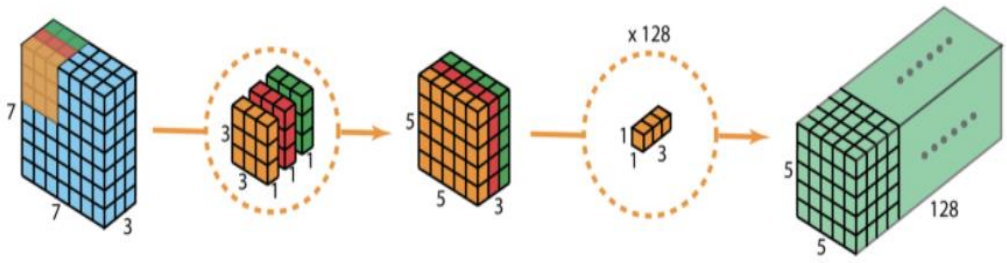


Figure 28: Block of Depth-Wise Separable Convolution

Figure 28 illustrates the general mechanism of depth-wise separable convolutions. With the conventional method if we had 7x7x3 dimensional feature map we would need 128 kernels with 5x5x3 dimension in order to output 5x5x128 dimensional feature map. However, in depth-wise separable convolutions we divide convolution operation into two parts, depth-wise and point-wise convolutions. Firstly, using 3 kenels with 3x3 dimension each the method depth-wise convolves filters from input feature map separately and afterwards concatenate them, resulting in 5x5x3 dimensional immediate result. The immediate result afterwards point-wise convolved with 128 kernels with dimension 1x1x3 to get final output of 5x5x128 dimensional feature map.

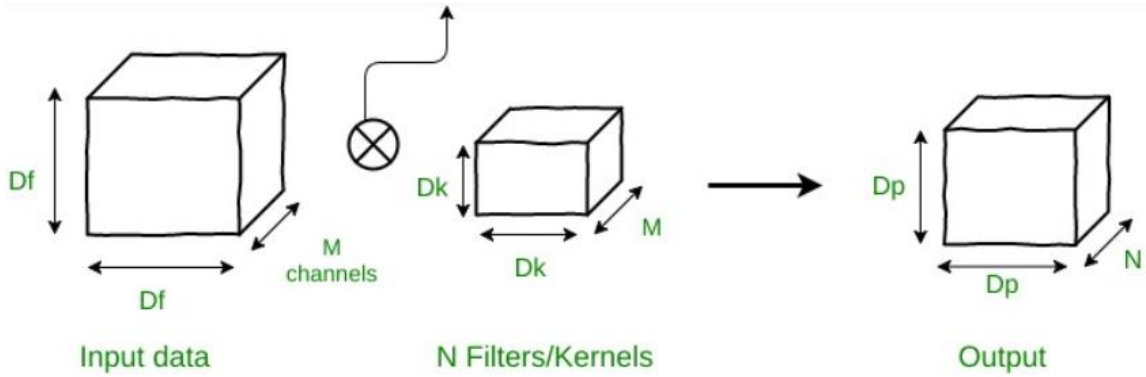


Figure 29: Conventional CNN Block

To be precise, let us compare the complexities of these two convolution methods. Usually, the number of calculations in one convolutional operation is defined to be equal to the size of filter used

$$O(c) = Dk \times Dk \times M$$

We get the number of multiplications needed for one convolutional operation by multiplying the dimension of used filter for the conventional convolutional method. Note that we have N number of filters used for the overall convolutional operation as illustrated in figure 29. So, we can easily find the total number of convolutional operations since each filter will slide Dp horizontally and vertically to get feature map with DPxDPxN dimension.

$$O(n) = Dp \times Dp \times N$$

So multiplication of number of convolutional slide operation with number of multiplication operations needed for each convolution would give us the final complexity of conventional CNN layer.

$$O(Tc) = O(c) \times O(n) = Dk^2 \times Dp^2 \times N \times M$$

O(T) is total number of multiplications needed in each layer of conventional Convulitional Neural Network. All elements and dimensions used in the calculation are described in figure 24. Now let us calculate the complexity for one layer of depth-wise separable Convolution. As we mentioned before first part of such operation is called depth-wise Convolution operation

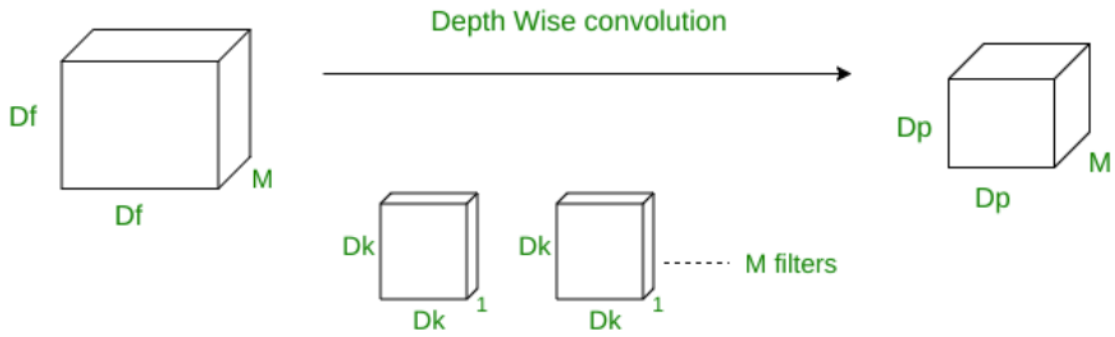


Figure 30: Depth-Wise Convolution Operation

Since we are using Dk x Dk x 1 filters are convolved separately with M number of input feature map channels each convolutional operation consists of Dk x Dk multiplications. Considering that each filter would slide Dp times both horizontally and vertically, following number of multiplications is needed for overall depth-wise convolution.

$$O(d) = M \times Dp^2 \times Dk^2$$

Now that we have number of multiplications for depth-wise convolutional operation, let us calculate it for point-wise convolutional operation as to obtain final complexity of depth-wise seperable Convolution.

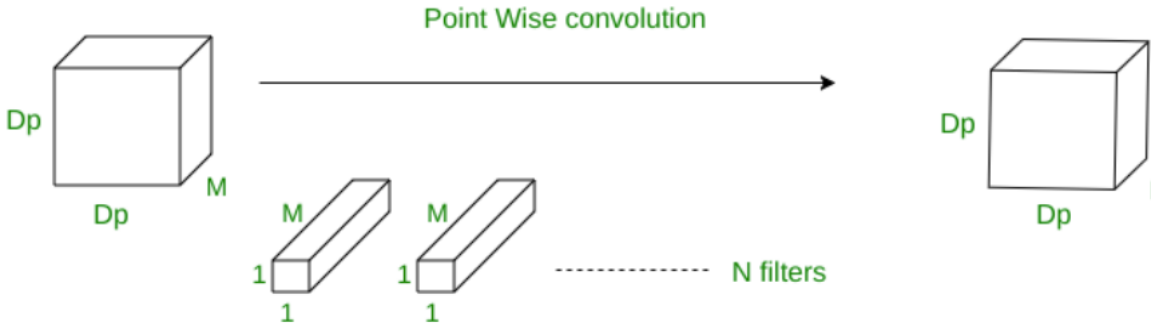


Figure 31: Depth-Wise Convolution Operation

For point-wise convolutional operation, single operation takes 1x1xM = M multiplications. The filters are again slides by Dp times both horizontally and vertically for N dimension.

$$O(p) = 1 \times 1 \times M \times Dp \times Dp \times N = M \times N \times Dp^2$$

Finally, by adding these two number of mutiplications we get overall complexity for one block of depth-wise seperable Convolution.

$$O(Td) = M \times Dp^2 \times Dk^2 + M \times N \times Dp^2$$

Now that we have both complexities let us compare the difference in their robustness.

$$O(Tc) = Dk^2 \times Dp^2 \times N \times M = M \times Dp^2 \times N \times Dk^2$$
$$O(Td) = M \times Dp^2 \times Dk^2 + M \times N \times Dp^2 = M \times Dp^2 \times (N + Dk^2)$$
$$R = O(Tc) / O(Td) = N \times Dk^2 / (N + Dk^2)$$

With this given difference in complexity for example, if we had output feature map with 512 filters and kernel size with 7x7 then the difference would be equal to

$$R = 512 \times 7^2 / (512 + 7^2) = 44.72$$

In this given example depth-wise seperable Convolution operation is 44.72 times faster than standard Convolution operation. It is also important to note that deeper in the CNN architecture this complexity ratio increases rapidly due to the increasing number of channels in feature maps towards the end. Sandler et al. integrates this idea of depth-wise sperable Convolution layers into their main building block of architecture, called residual and linear bottlenecks [20]. The idea behind deisgning model based on bottleneck layers is to gain ability of encoding feature maps in low dimensional subspaces. The final convolutional operation in bottleneck is a linear activation layer, since having non-linear activation results in information loss. Due to the low dimensions in feature maps MobileNetV2 designed in a way to minimize information loss.

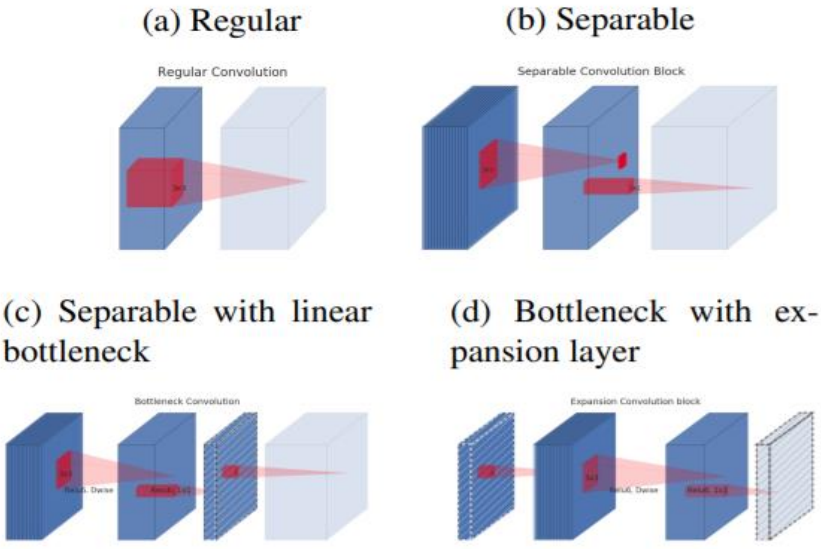


Figure 32: Evolution of Inverted Residual and Linear Bottleneck

Figure 32 illustrates the though process of designing inverted residual and linear bottleneck layers. Also, note that gray block in figure 26 implies the start of the next bottleneck. We have already mentioned the differences between regular and separable convolutional layers shown in 26(a) and 26(b). Figure 26(c) already discussed the use for the final operation in the bottleneck is activated linearly. Note that final operation is pointwise convolutional operation using k' number of filters with 1x1xRk dimension. The block demonstrated in 26(D) adds another point-wise convolutional operation to the start of the bottleneck layer. The main responsibility of this operation is to expand the dimension of the feature map with the factor of R from k to R*k, hence, the name expansion layer. The next layer is depth-wise convolutional operation with 3x3xRk filter followed by point-wise operation activated linearly. Note that expansion and depth-wise convolution layers use ReLU6 which is a non-linear activation function.

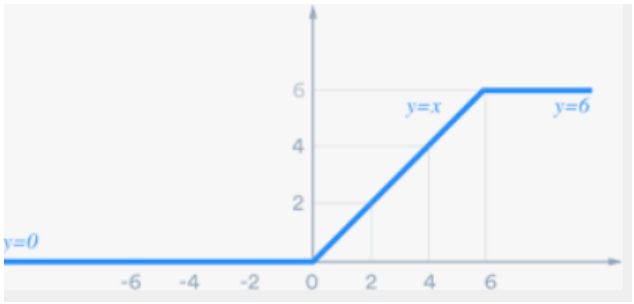


Figure 33: ReLU6 activation function

ReLU6 activation function is rectified linear activation function in which the maximum value after the activation is 6.

$$ReLU6(x) = \begin{cases} min(x,6), & if\ x > 0 \\ 0 & otherwise \end{cases}$$

Here is the ReLU6 function for further explanation and simplicity.

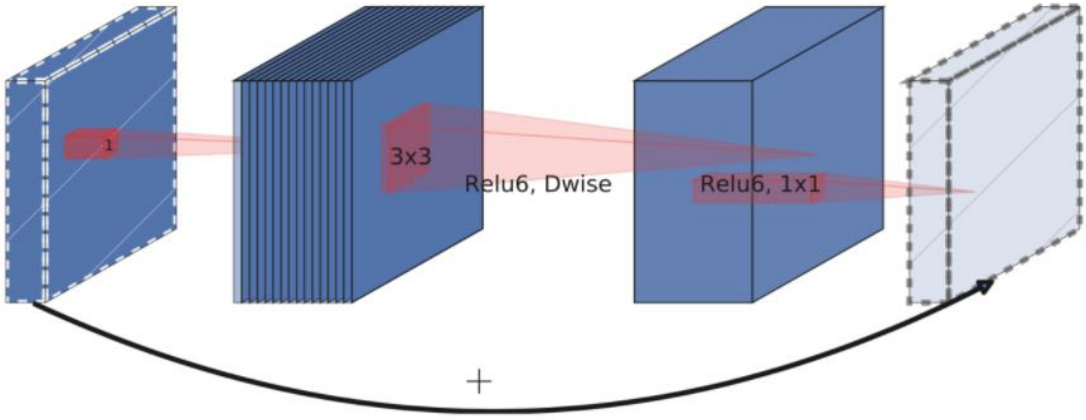


Figure 34: Visualization of Inverted Residual and Linear Bottleneck

Figure 34 describes the final version of Inverted Residual and Linear Bottleneck used in the design of MobileNetV2 architecture. The difference from the block shown in 26(c) is that now we have residual skip connections to the input of next bottleneck. Please, note that residual connections are only used whenever the input feature maps of the bottlenecks has matching dimensions.

Table 3: Details of Each Layer used in Inverted Residual and Linear Bottlenecks

Input	Operation	Output
d x d x k	1x1, point-wise + ReLU6	d x d x Rk
d x d x Rk	3x3 depth-wise, (stride = s) + ReLU6	d/s x d/s x Rk
d/s x d/s x Rk	1x1, point-wise, linear	d/s x d/s x k'

Let us summarize the mechanism of the bottlenecks used in MobileNetV2. First, expansion layer of 1x1 filters used to increase number of channels by the factor of R. Depth-wise convolutional operation with 3x3 filters used with ReLU6 activation which reduces the dimension of each channel by the stride value s. Thirdly, point-wise convolutional operation that changes channel dimension of feature map to the number of k'. Finally, residual skip connections are added wherever dxd dimensions of the feature maps matches.

Table 4: Original Architecture of MobileNetV2 Model Used in our ANPR System

Input	Operator	R	k	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

We present the details of the architecture of MobileNetV2 model in table 4. R refers to the expansion factor, k is the dimension of the next output feature map, n is the filter dimensions nxn, and finally, s is the value of stride use. Based on table 4, residual connection were implemented between feature maps in layers 2-3, 6-7, and 8-9.

We have already discussed the major design decisions for our Automatic Number Plate Recognition System. Now let us discuss our overall design architecture for ANPR system and how to drive meaningful investigations based on the system pipeline.

2.3 ANPR System Pipelines

Since we have two main methodologies in order to perform Optical Character Recognition We do need to further discuss the mian building blocks for 2 versions of the design architecture proposed.

For the implementation of Automatic Number Plate Recgonition system we have first discussed using addition YOLOv4 model in order to detect and classify characters as an end-to-end deep learning solution as illustrated in Figure 35(a). On the other hand, The second pipeline has an Optical Character Recognition solution based on intensive preprocessing and character segmentation, followed by character classification with MobileNetV2 model.

Even though both pipelines include YOLOv4 object detection model in order to locate licanse plates from raw traffic camera feed, OCR parts of the pipelines are geniunely dissimilar with each-other. Also it is necessary to mention that we for Plate Detection model we will be also trying tiny version of YOLOv4 which is distinguished by its reduced backbone architecture. Indeed, tiny YOLOv4 cannot perform any better than YOLOv4 model itself, however, given its much higher speed of inference it is quite important to consider this model further. Tiny YOLOv4 model's excessive inference speed is crucial to reduce processing costs of an ANPR system in real life, thus, making it a viable alternative in case of similar performance to vanilla YOLOv4 model.

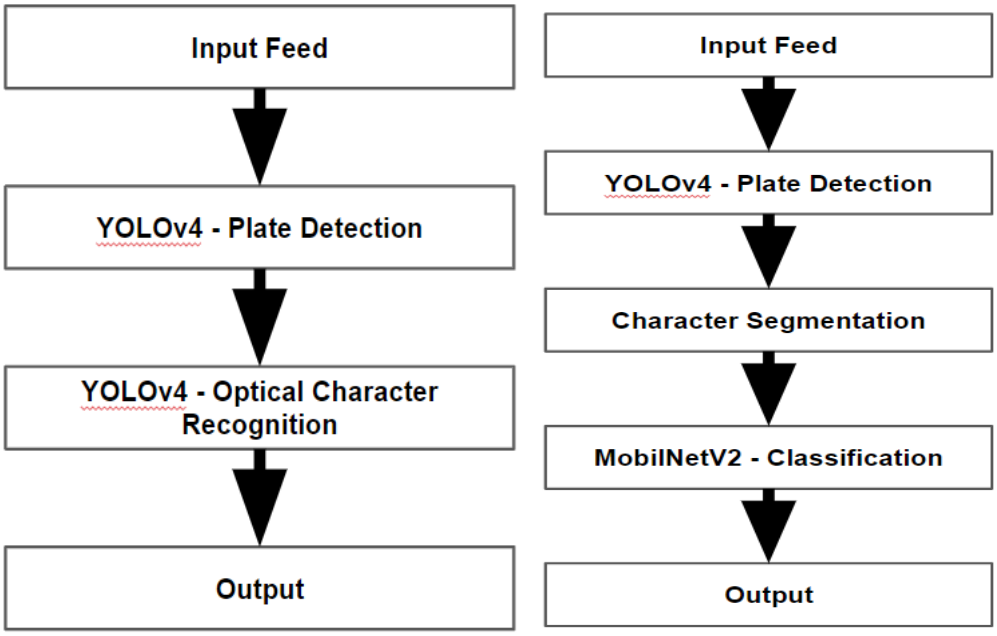


Figure 35: Proposed ANPR Systems (a) First Proposed Pipeline With YOLOv4 as OCR Solution (b) Second Proposed Pipeline With Character Segmentation and Classification

Since we are proposing 2 different pipelines we will need to make an excessive comparisons between them by conducting different experiments. In the next section, this paper will include the comprehensive experiments investigating the static and dynamic metrics of the proposed pipelines. We will also discuss differences of YOLOv4 and tiny YOLOv4 in depth as license plate detection models. However, before moving onto the next section we need to discuss the training data used for ANPR systems since the overall design includes several deep learning models.

2.4 Data

As we are using 2 deep learning algorithms in our ANPR system, namely YOLOv4 and MobileNetV2 models, we will discuss individual datasets that have been gathered for these model's training process. But first, let us examine the domain of the ANPR system that we are building. The final ANPR system produced by the paper will be used to identify vehicle registration numbers issued by the Republic of Azerbaijan. It is quite fortunate that Azerbaijani plate numbers follow set of numerous rules, even though there are 2 primary version of license plates available.

Figure 36: Visualization of Vehicle Registration Plate Templates Issued by the Republic of Azerbaijan

As we mentioned before, Azerbiajani license numbers follow specific set of rules including, but not limited to plate dimensions, character font type, and set of allowed characters. The most

prominent types of license plate are 2 templates which has white background. Even though, there are other possible background colors, since it is quite easy task to remove them by masking due to monotone plane background. Also notice that "AZ" text with Azerbaijani flag logo also located in the leftmost side of each license plate. Due to the fixed location of this text it can be easily removed from the license plate by rule-based algorithm in order to run OCR algorithms without noise. The license plates mostly follow the patter of "NN - LL - NNN" where the "N" and "L" stands for number and letter characters respectively. Azerbaijani license plates also allow only numbers from 0 to 9 and fixed size of Latin alphabet characters. Set of characters allowed in the vehicle registration plates are

"ABCDEFGHIJKLMNPQRSTUVWXYZ0123456789"

Note that only 25 Latin alphabet characters included since the characters follow the universally adopted Latin alphabet characters for license plates, and the vehicles does not include character "O" since letter "O" and number "0" are issued as exactly the same due to the selected character font type.

Moving on, we have gathered training data for YOLOv4 object detection model from 2 major sources in order to custom train the YOLOv4 and tiny YOLOv4 models for the purposes of this paper. The first data source is Google Open Images Dataset version 6, which stores over 15 million images overall on 600 different categories. One of the categories include vehicle registration plates with bounding box annotations perfectly suitable for our training purpose.

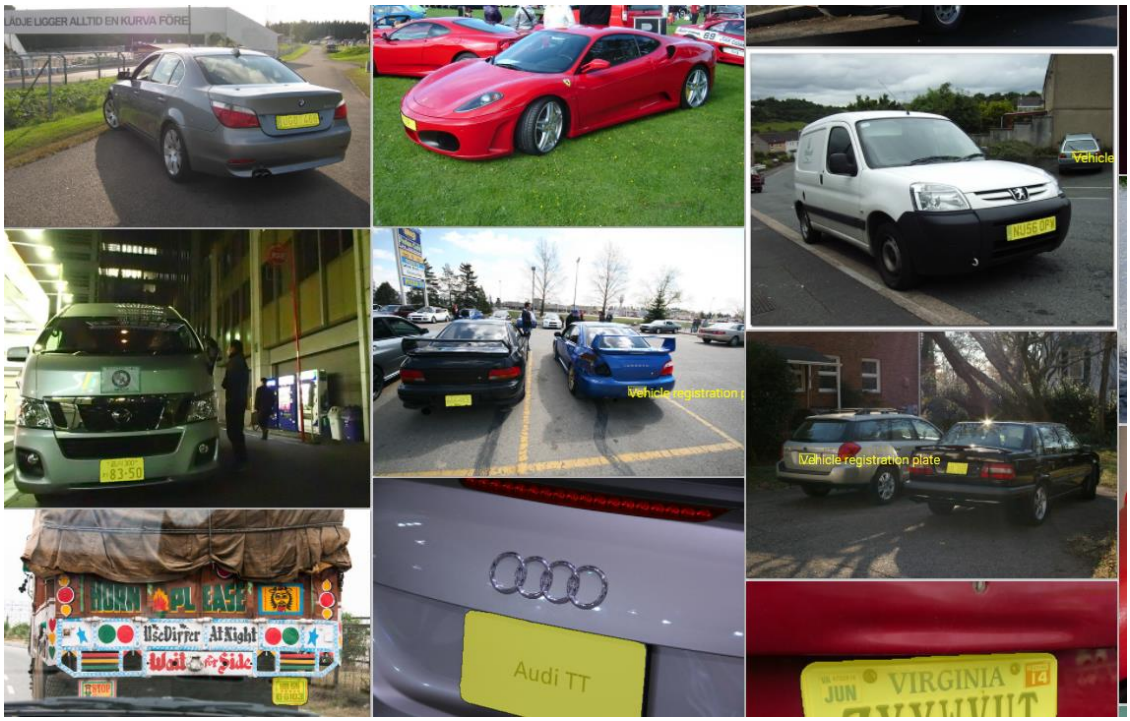


Figure 37: Visualization of Image samples gathered from Google Open Images V6

For training our object detection on locating license number plates we have gathered over 10000 images from the aforementioned data source. Figure 37 illustrates some of those training samples obtained from the data source. These images provide license plates in different shape from various angles all over the world which, and thus, making it perfectly appropriate dataset in order to generalize our object detection models. However, since we are working on a specific domain we also needed another data source where we can obtain training samples homogenous with the intended domain.

Therefore, we selected images captured by CCTV road surveillance cameras as our second data source. These traffic cameras provided us with around 10000 images of cars with license plate too. After gathering such training samples it was possible to train our YOLOv4 and tiny YOLOv4 models appropriately and obtain generalized accurate object detection models. For training object detection models we have also employed transfer learning with pre-trained YOLOv4 and tiny YOLOv4 weights trained on COCO dataset [25].

Figure 38: Visualization of Image Samples Gathered from CCTV Road Surveillance Cameras

As it is illustrated in figure 38, this dataset also contains images from different angles, usually from mounted perspective since the data source is CCTV road surveillance cameras. It is also observable that apparently almost all of the vehicle registration plates issued by the Republic of Azerbaijan are from the templates in white background as demonstrated in the figure 3. Therefore, the same pattern is also applicable to the real data from road surveillance domain.

Unlike the data needed for object detection algorithms, character classification requires particularly less extensive data.

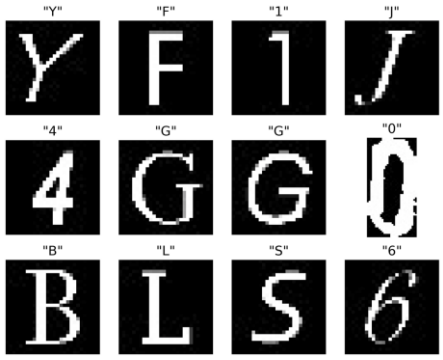


Figure 39: Visualization of Data Samples Used in MobileNetV2 Character Recognition

In fact, we only need binary images of segmented Latin alphabet characters for each category. It is also important to note that we only have 35 categories for character classification, and therefore, training samples for letter 'O' and number '0' are combined into one category. We have collected binary segmented images of both digital and handwritten characters for MobileNetV2 character classifier. Some training samples for character classification problem are illustrated on figure 39. We have gathered from 250 to 300 samples for each character totalling up to around 7000 images.

Since we have another YOLOv4 model implemented as Optical Character Recognition algorithm, this required training samples too. YOLOv4 is an object detection algorithm, and, thus requiring annotated images with bounding boxes as opposed to binary segmented character images. Considering that we have two data sources for license plate detection full of vehicle registration numbers, we have decided to extract several license plates from these two data sources and use their contents to feed into our YOLOv4 object detection model for solving OCR problem. We have extracted and manually annotated 1000 license plates from Google Open Images V6 dataset and 7000 vehicle registration plates from images captured by CCTV road surveillance cameras in order to use as training data for our YOLOv4 OCR model, totalling to 8000 license plate images with bounding box annotations for letter and number characters.

Table 5: Data Sources and Training Dataset Description.

Data Sources			
Source	Description	Usage	N of samples
Google Open Images V6	Category: Vehicle Registration Plate	License Plate Dection	11238
CCTV	feed from road surveillance cameras	License Plate Detection	9876

Characters dataset	handwritten + digital	Character Classification	7043
Deep Learning Models			
YOLOv4 - Object Detection		LPR	21114
tiny YOLOv4 - Object Detection		LPR	21114
YOLOv4 - Optical Character Recognition		OCR	8000
MobileNetV2 - Character Classification		OCR	7043

Table 5 demonstrates the details of data sources and the description of how each model uses these training datasets. We will further discuss how we conduct experiments on the models trained on aforementioned datasets and what test samples used for investigations in the next chapter.

3. EXPERIMENTS AND EVALUATION

In this section we will investigate throught our implementations and try to determine best architectural pipeline for ANPR system based on our empirical evidence. Firstly, we will look at differences in object detection models resulted by training YOLOv4 and tiny YOLOv4 algorithms. Then we will discuss the performances of the Optical Character Recognition models namely YOLOv4-OCR and segmentation + MobileNetV2 classification. At the end, to deduce clear differences and distinguish the best design, we will comprehensively investigate the performances of the overall pipelines as a whole. However, before startinig in depth analysis, we need to define on what basis we are going to evaluate each model. Note that, for all the experiments, we are using RTX 2080 Super Max-q model GPU.

3.1 Definition of the Metrics

Other than dynamic metrics defined for each model and complete pipeline, we need to consider static metric for inference speed of the algorithms. Note that if the overall pipeline cannot achieve at least 30 frame per second inference speed, then the implementation of such architectural design is unfeasible for real-worl application of ANPR system. Therefore, we will define our static metric as frame per second that is calculated by

$$M(s) = 1\ sec\ /\ It(s)$$

Where M(s) is the static metric and It(s) is the time it takes the algorithm to process one input image. Now that we have defined our static metric we can move on to dynamic metrics that will be used in this section. Even though in for Machine Learning models we prefer using dynamic metrics with such as accuracy score, F-score, and ROC AUC, it is preferable to use mean average precision metric for object detection models [26].

$$mAP = 1/C \sum^{C}_{k=1} AP_k$$

mAP metric refers to the mean of Average Precisions over C classes.

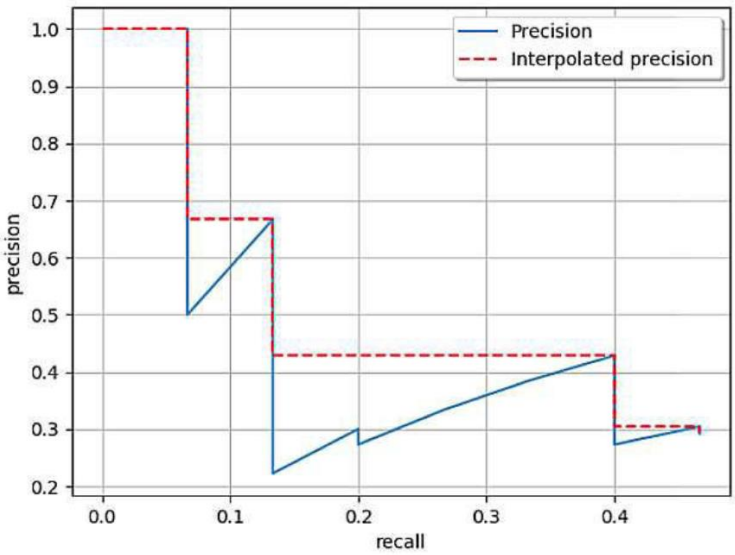


Figure 40: Example of Interpolated Precision - Recall Curve

Average precision, on the other hand, is defined as the average of the Area Under the Interpolated Precision and Recall Curve.

$$AP = 1/n \sum_{Ri} (Pmax_i * R_i)$$

Average precision is calculated based on this equation where n refers to the number of thresholds used in Precision-Recall curve, Pmax defined as Interpolated Precision, and R referring to the Recall value. In case of license plate detection, indeed, we are calculating only average precision since we only have one class for the object detection problem. However, for the sake of clarity, we will accept metric for this part as mAP by convention. In terms of YOLOv4 as object detection algorithm we will be using mean average precision metric where we have C = 35 classes of characters as different objects.

$$Acc = (TP + FP) / (TP + TN + FP + FN)$$

For MobileNetV2 model we will use accuracy score as metric since it is a classification problem. Finally, we need to define an accuracy metric to evaluate the performance of complete pipeline. However, when if difference performance metrics used for OCR problem, it becomes unfeasible to compare YOLOv4-OCR and MobileNetV2 models, therefore, we will also calculate accuracy score for YOLOv4 model by giving test samples of images where only one character is visible.

There is a need to define one more metric in order to measure the performance of pipelines overall. In that case, we will be using modified version of Accuracy score for this purpose where the top 1 result is expected to match the ground truth of the full text.

$$Acc(T) = \text{Number of Hits} / \text{Number of Total Cases}$$

Where for each test case the algorithm receives 1 if the full extracted text is exact match with the ground truth value of license plate number, and it receives 0 for all other cases even the algorithm misses with 1 character mismatch. The reason is we are treating our final proposed ANPR system to be usen in real world application and if the output of the system misses even with only 1 character, then the obtained result is has no real use. Now that we have defined our performance metrics let us one by one examine our models developed for Automatic Number Plate Recognition.

3.2 Evaluation of Plate Detection Models

As we mentioned before we are using YOLOv4 and tiny YOLOv4 object detection models in order to locate license plates from the images provided by traffic cameras. For evaluation we will be using IOU thresholds of 0.5,0.6, and 0.65 for both YOLOv4 and tiny YOLOv4 models in order to be able to compare the models thoroughly. For testing, we are using 500 sample images taken from CCTV road surveillance cameras.

Table 6: Evaluation of License Plate Detection Models

Model	Average FPS	IOU-thresh	mAP
YOLOv4	33	0.5	0.843
tiny YOLOv4	125	0.5	0.652
YOLOv4	34	0.6	0.818
tiny YOLOv4	127	0.6	0.541
YOLOv4	34	0.65	0.802
tiny YOLOv4	129	0.65	0.509

Looking at the details of our assessment for Object Detection Problem, it is clear that YOLOv4 model has much better performance in terms of mean average precision. In fact, normally, models trained on YOLOv4 architecture is expected to have a mAP score around 60

to 70. However, since we only have one class for object detection, the obtained results are fairly expected. On the other hand, tiny YOLOv4 model seems to struggle in detecting vehicle registration plates. During our comrehensive assesment, it became clear that tiny YOLOv4 model specifically struggled locating license plates that are far back from the perspective of camera. By further exploring the these test cases for tiny YOLOv4 model, we found out that the reason is that the model cannot properly detect smaller license plates located in the far end of the road due to very small confidence level for such detections caused by low quality imaging. It is also important to consider that feature map dimensions of tiny YOLOv4 is quite small, resulting in failure to detect smaller objects. However, the model still works well for large license plates and the ones fairly close to the camera. Putting aside the poorer performance, tiny YOLOv4 astonishingly processes images nearly 4 times faster than YOLOv4 model. Therefore, in some real world cases where processing resources are limited tiny YOLOv4 might be even better option to employ in an ANPR system. We also, see that with increasing IOU threshold both models start to show worse performance. But it is more apparent for tiny YOLOv4 model when we increase IoU threshold from 0.5 to 0.6.

3.3 Evaluation of OCR models

Since image segmentation is based on processing techniques and contour detection as opposed to Deep Learning models used in ANPR system we do not have explicit way of evaluating this part of the algorithm. However, we will be able to implicitly evaluate the performance of this character segmentation when comparing complete pipelines to each other as character segmentation is one of the core contribtutors to the success of ANPR system. Therefore, only YOLOv4-OCR and MobileNetV2 models will be evaluated in this subsection. note that for mAP performance we have 500 test cases of license plates taken from the output of YOLOv4 license plate detection model with IOU threshold of 0.5. For accuracy scores, we are using 500 images of single character images cropped out of vehicle registration plates for YOLOv4-OCR and same 500 images of segmented characters for MobileNetV2.

Table 7: Evaluation of OCR models of ANPR system

Model	Average FPS	Metric	Score
YOLOv4 - OCR	34	mAP	31.9
YOLOv4 - OCR	36	Acc	79.0
MobileNetV2	93	Acc	96.6

First noticable detail from the evaluation of OCR models is that YOLOv4 did not perform very well, in fact, the performance of YOLOv4 as OCR model is way below the expected outcome. However, MobileNetV2 showed an outstanding accuracy score of %96.6 for test cases.

Also, it is important notice MobileNetV2 runs with very low overhead with average inference speed of 93 frames per second. It is fortunate that at least one of the proposed algorithms did exceptionally well on the OCR problem. Note that Inference speed for MobileNetV2 also includes time needed for preprocessing of image and segmentation algorithm. Unfortunately, YOLOv4 model did not produce any meaningful results. There are several nuances we need to consider in order to explain these results. First of all, this YOLOv4 model is trained to detect and classify 35 different classes as opposed to single class used in license plate detection part. Having these many classes undeniably affects the overall performance. Moreover YOLOv4 is an object detection model by nature, thus, tuned to detect objects based on more general features of the objects. For example, end-to-end object detection models are usually used to differentiate objects such as Horse, Human, Airplane, and Car which has completely different general futures as far as object detection solutions are concerned. However, in this given problem we have 35 characters which presents too many similar futures in terms of object detection problem. To mention few, all characters are roughly the same size, they are all vector objects, have same proportion, similar interest points, and even they all have the same color distribution. All of this factors might have contributed to the poor performance observed by YOLOv4 model in solving Optical Character Recognition problem. We have seen the clear performance gap between character segmentation + MobileNetV2 character classification technique and YOLOv4-OCR solution. However it is important to complete our experiments by evaluating the overall performance of the proposed ANPR pipelines.

3.4 Evaluation of Proposed ANPR Systems

As we mentioned in sub-section 3.1, we will use our modified accuracy score to evaluate final outcomes of the proposed ANPR systems. Here we consider again 500 images captured by CCTV road surveillance cameras. However, to get precise assessment on ANPR implementations we will only consider license plates captured from closer perspective to the cameras. To be specific, we will only account license plates detected in the bottom %40 of each frame towards the modified accuracy score.

$$f(y) \in T \text{ , iff } C(y) < 0.4*H$$

where f(y) refers to the output of ANPR system, T refers to the results counted towards accuracy score, C(y) is the vertical center point coordinate of input license plate image, and H refers to te height of the image. Moreover we exclude, license plates that are not detected by object detection or partially detected where the whole ground truth text is not available. Note that some input images contain more than single detected license plate. IoU threshold is equal to 0.5.

Table 8: Evaluating Overall Performance of Proposed ANPR System

Model Description	Average FPS	mod. Accuracy Score
YOLOv4 + YOLOv4-OCR	31	0.546
tiny YOLOv4 + YOLOv4-OCR	31	0.521
YOLOv4 + MobileNetV2	33	0.951
tiny YOLOv4 + MobileNetV2	91	0.933

As expected, both architectures that include YOLOv4 model as Optical Character Recognition solution demonstrates quite poor performance with a accuracy score slightly better than 0.5. On the other hand, the other 2 proposed ANPR systems show promising results. Upon further investigation we do notice that the model using tiny YOLOv4 has %1.8 lower accuracy score than the pipeline using YOLOv4 model. This probably happened because tiny YOLOv4 model is slightly worse at localizing bounding boxes, thus causing the obtained license plate to have more noise on the boundaries of the image. In this case, segmentation algorithm that we are running might have difficulty to detect all the given character segments, thus, cauasing lower accuracy score. However, We can see that the ANPR build upon tiny YOLOv4 + MobileNetV2 architecture is almost 3 times faster than the other well performing ANPR system.

4. CONCLUSION

Unfortunately, the use of YOLOv4 model to solve Optical Character Recognition problem did not produce any fruitful results. On the other hand, Based on conducted experiments and emprical evidence we can comfortably conclude that the use of character segmentation and classification combination in ANPR system has crucial contributions to the overall success of ANPR system. Especially, character classficiation algorithm based on MobileNetV2 model architecture proved to be useful for such problems. Indeed, using MobileNetV2 architecture ANPR system was able to perform at astonishing accuracy rate of %95.1 with YOLOv4 license plate detection algorithm. The same classification model enabled us to obtain quite robust and accurate ANPR pipeline with %93.3 accuracy and 91 frames per second inference speed using tiny YOLOv4 vehicle registration detector. Both ANPR systems can be deployed in production level depending on the user's requirements and preferences.

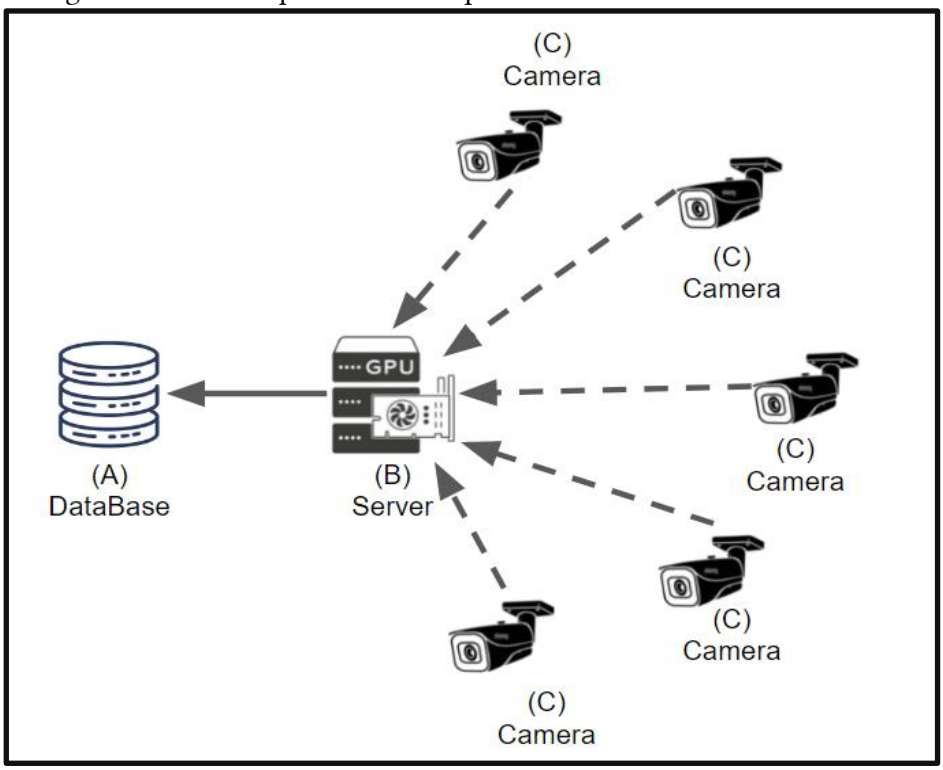


Figure 41: Deployment Diagram of Real-life Implementation of ANPR System

The proposed ANPR systems are considered to be minimal viable version of the potential system. As we already mentioned both system designs demonstrates satisfactory results. ANPR system containing the combination of YOLOv4 License Plate Detector + MobileNetV2 character classifier has the best accuracy score. However, we should keep in mind that the processing rate of this pipeline is only 33 frames per second and can be considered adequately costly module. It is possible to employ this pipeline in the deployment of full-scale ANPR system as illustrated in figure 35. But considering the overhead of this design using high-end server with gpu or tpu processing capabilities would be adequate. As shown in figure 41 the main idea of the system would be that traffic control cameras would forward send their to the high-end server where the received input would be processed by our proposed design to obtain output later to be stored in the database. This kind of system would also support deployment of the other ANPR design we discussed where we emply tiny YOLOv4 + MobileNetV2 model combination.

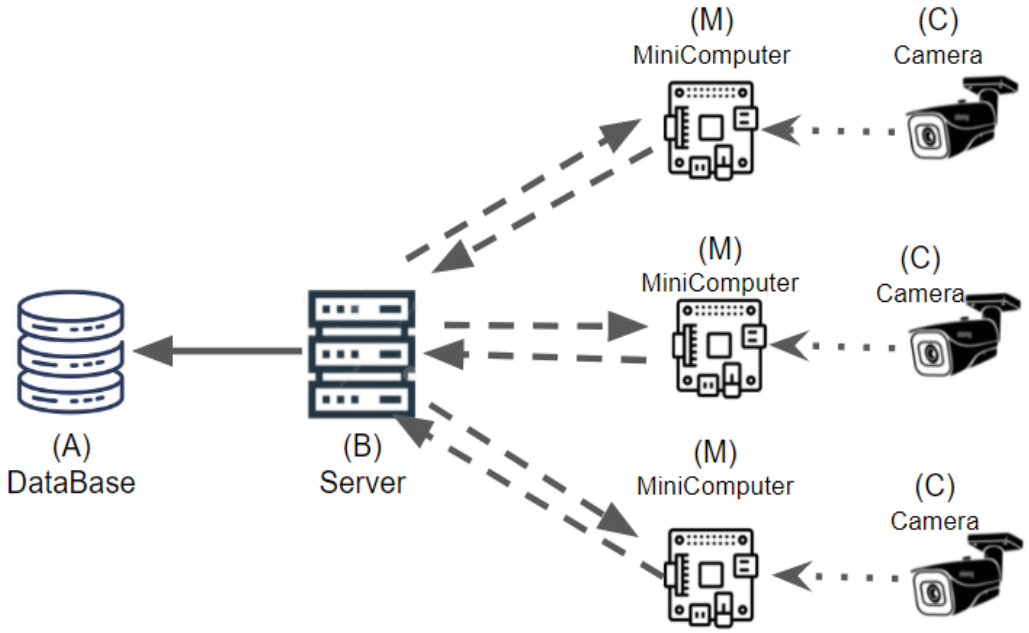


Figure 42: Alternative Deployment diagram of Real-life Implementation of ANPR System

However, considering the low overhead and robustness of this proposed design it would be much more sensible option to use this module as illustrated in figure 42. Since this pipeline has very low overhead the ANPR processing can be transferred mobile minicomputers which would process feed from CCTV cameras and forward the output of our ANPR algorithm to the remote server. In this case, we can use an intermediate server as the ANPR operations requiring high processing power is moved to somewhere else. The server is only needed as governing agent for data transfer and master node to manage, and occasionally update minicomputers which are responsible for ANPR processing. The system architecture design described in

figure 41 is much cheaper than the formerly proposed deployment module and much more scalable. Having adequately discussed the real world applications of our best performing ANPR designs, we can now summarize our review to move on to the challenges and future work.

4.1 Future Work and Notes

It is rather unfortunate that the idea of solving OCR in other new ways by emplyong Object Detection Algorithm did not perform well. Having said that this should be inspiration, rather than discouragement, to propose novel ANPR designs in order to acquire even better performing systems. Currently, deep learning is ever-growing field where we are introduced to new novelties and state-of-art concepts quite frequently. Considering the advancements in technological field with increased availability of higher processing power it is safe to say that ANPR systems are bound to be evolved into employing end-to-end deep learning architectures very soon. Even today's best performing ANPR systems can find themselves vulnerable when encountered by extremely noisy or low quality imaging where simple processing operations cannot perform adequately any longer. Perhaps, use of attention models or two stage object detectors can prove to be useful in the domain of Automatic Number Plate Recognition problem. Further it would be sensible idea to use object detection algorithms in order to segment characters within license plates to be later used by character classification algorithms. All in all, it is certain that Object detection algorithms are much more effective in detecting objects in noisy or low quality environment as opposed to the rule based solutions.

REFERENCES

[1] Lubna, Naveed Mufti, and Syed A.A. Shah. 2021. "Automatic Number Plate Recognition:A Detailed Survey of Relevant Algorithms" *Sensors* 21, no. 9: 3028. https://doi.org/10.3390/s21093028

[2] Pankaj Mukhija and Pawan Kumar Dahiya. 2021. *Challenges in Automatic License Plate Recognition System: An Indian Scenario.* Conference: Fourth International Conference on Computational Intelligence and Communication Technologies (CCICT)-2021. IEEE. Sonepat,India. DOI:10.1109/CICT53244.2021.00055

[3] Abhipsa S Kalyanshetti, K Hemanth Ratna Kumar, Chethan R, Karthick R. 2020. *Automatic License Plate Recognition using MSER and CNN.* INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) NCETESFT – 2020 (Volume 8 – Issue 14). Bengaluru, India
https://www.ijert.org/automatic-license-plate-recognition-using-mser-and-cnn

[4] Shyang-Lih Chang, Shyang-Lih Chang, Sei-Wang Chen, and Li-Shien Chen. 2004. *Automatic License Plate Recognition.* IEEE Transactions on Intelligent Transportation Systems 5(1):42 – 53. DOI:10.1109/TITS.2004.825086

[5] Hasan Erdinc Kocer and Kerim Kürşat Çevik. 2011. *Artificial neural networks based vehicle license plate recognition.* Procedia Computer Science Volume 3, 2011, Pages 1033-1037. DOI: 10.1016/j.procs.2010.12.169

[6] Zhenbo Xu, Wei Yang, Ajin Meng, Nanxue Lu, Huan Huang, Changchun Ying, and Liusheng Huang. 2018. *Towards End-to-End License Plate Detection and Recognition: A Large Dataset and Baseline.* Computer Vision – ECCV 2018. ECCV 2018. Lecture Notes in Computer Science, vol 11217. Springer, Cham. https://doi.org/10.1007/978-3-030-01261-8_16

[7] Alina Haines. 2009. *The role of automatic number plate recognition surveillance within policing and public reassurance.* Doctoral thesis, University of Huddersfield.
https://www.researchgate.net/publication/277177684_The_role_of_automatic_number_plate_recognition_surveillance_within_policing_and_public_reassurance

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. *Rich feature hierarchies for accurate object detection and semantic segmentation.* CVPR 2014
https://arxiv.org/abs/1311.2524

[9] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. 2015. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* Advances in Neural Information Processing Systems 28 (NIPS 2015) https://doi.org/10.48550/arXiv.1506.01497

[10] Ross Girshick. 2015. *Fast R-CNN.* ICCV https://arxiv.org/abs/1311.2524

[11] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. *You Only Look Once: Unified, Real-Time Object Detection.* https://doi.org/10.48550/arXiv.1506.02640

[12] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. 2020. *YOLOv4: Optimal Speed and Accuracy of Object Detection.* Computer Vision and Pattern Recognition (cs.CV); Image and Video Processing (eess.IV) https://doi.org/10.48550/arXiv.2004.10934

[13] Joseph Redmon, Ali Farhadi. 2016. *YOLO9000: Better, Faster, Stronger.* Computer Vision and Pattern Recognition (cs.CV). https://doi.org/10.48550/arXiv.1612.08242

[14] Joseph Redmon, Ali Farhadi. 2018. *YOLOv3: An Incremental Improvement.* Computer Vision and Pattern Recognition (cs.CV). https://doi.org/10.48550/arXiv.1804.02767

[15] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh. 2019. *CSPNET: A NEW BACKBONE THAT CAN ENHANCE LEARNING CAPABILITY OF CNN.* Computer Vision and Pattern Recognition (cs.CV). https://doi.org/10.48550/arXiv.1911.11929

[16] Svetlana Lazebnik, Cordelia Schmid, Jean Ponce. 2006. *Beyond bags of features: spatial pyramid matching for recognizing natural scene categories.* IEEE Conference on Computer Vision & Pattern Recognition (CPRV '06), New York, United States. pp.2169 - 2178. https://inc.ucsd.edu/mplab/users/marni/Igert/Lazebnik_06.pdf

[17] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, Jiaya Jia. 2018. *Path Aggregation Network for Instance Segmentation.* 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). The Chinese University of Hong Kong, Peking University, SenseTime Research. Pages: 8759-8768. https://arxiv.org/pdf/1803.01534.pdf

[18] Diganta Misra. 2020. *Mish: A Self Regularized Non-Monotonic Activation Function.* ***Machine Learning (cs.LG);*** Computer Vision and Pattern Recognition (cs.CV); Neural and Evolutionary Computing (cs.NE); Machine Learning (stat.ML). https://arxiv.org/abs/1908.08681

[19] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, Dongwei Ren. 2019. *Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression.* School of Mathematics, Tianjin University, China 2College of Intelligence and Computing, Tianjin University, China School of Information Technology and Cyber Security, People's Public Security University of China https://arxiv.org/pdf/1911.08287.pdf

[20] Mark Sandler Andrew Howard Menglong Zhu Andrey and Zhmoginov Liang-Chieh Chen. 2018. *MobileNetV2: Inverted Residuals and Linear Bottlenecks.* The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4510-4520
https://arxiv.org/abs/1801.04381

[21] Franc ̧ois Chollet. 2017. *Xception: Deep Learning with Depthwise Separable Convolutions.* 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) https://doi.org/10.48550/arXiv.1610.02357

[22] Estevao Gedraite and Murielle Hadad. 2011. *Investigation on the effect of a Gaussian Blur in image filtering and segmentation.* Proceedings ELMAR-2011, 2011, pp. 393-396. Zadar, Croatia https://ieeexplore.ieee.org/document/6044249

[23] Tarun Kumar and Karun Verma. 2010. *A Theory Based on Conversion of RGB image to Gray image.* International Journal of Computer Applications (0975 – 8887) Volume 7– No.2
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.206.4410&rep=rep1&type=pdf

[24] Ravi Srisha and Am Khan. 2013. *Morphological Operations for Image Processing: Understanding and its Applications.* Conference: National Conference on VLSI, Signal processing & Communications.
https://www.researchgate.net/publication/272484795_Morphological_Operations_for_Image_Processing_Understanding_and_its_Applications

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár. 2015. *Microsoft COCO: Common Objects in Context.* In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham. https://doi.org/10.1007/978-3-319-10602-1_48

[26] Rafael Padilla, Wesley L. Passos, Thadeu L. B. Dias, Sergio L. Netto and Eduardo A. B. da Silva. 2021. *A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit.* Electrical Engineering Program/Alberto Luiz Coimbra Institute for Post-Graduation and Research in Engineering (PEE/COPPE), PO Box 68504, Rio de Janeiro 21941-972, RJ, Brazil https://www.mdpi.com/2079-9292/10/3/279