AUTOMATIC LICENSE NUMBER PLATE RECOGNITION SYSTEM

PROJECT REPORT

SUBMITTED BY

GANGA KRISHNAN.G

TKM20MCA-2018

TO

THE APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE

OF

MASTER OF COMPUTER APPLICATIONS

THANGAL KUNJU MUSALIAR COLLEGE OF ENGINEERING

KERALA

JULY 2022

DECLARATION

I undersigned hereby declare that the project report "**AUTOMATIC LICENSE NUMBER PLATE RECOGNITION SYSTEM**", submitted for partial fulfilment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of **Prof. Natheera Beevi M**. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in the submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

KOLLAM GANGA KRISHNAN.G

18/07/2022

THANGAL KUNJU MUSALIAR COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that, this report entitled "**AUTOMATIC LICENSE NUMBER PLATE RECOGNITION SYSTEM**" is a bonafide record of the work submitted by **GANGA KRISHNAN.G (TKM20MCA-2018),** under our guidance and supervision, in partial fulfillment of the requirements for the award of the Degree of **Master of Computer Applications** in **APJ Abdul Kalam Technological University.** This report in any form has not been submitted to any other University or Institute for any purpose.

Internal Supervisor Head of the Department External Examiner

ACKNOWLEDGEMENT

A successful project is a fruitful culmination of efforts by many people, some directly involved and some others indirectly, by providing support and encouragement. Firstly, I would like to thank the almighty for giving me the wisdom and grace for making my project a memorable one. I thank him for steering me to the shore of fulfillment under his protective wings.

I express my sincere gratitude to **Dr. T A Shahul Hameed**, Principal of T.K.M College of Engineering for giving me an opportunity to present my project. I would like to thank **Dr. Fousia M Shamsudeen**, Assistant Professor and Head of the Department, MCA, TKMCE, for her constant support and encouragement throughout the project work.

With a profound sense of gratitude, I would like to express my heartfelt thanks to my guide **Prof. Natheera Beevi M**, Department of Computer Applications, TKMCE, for her expert guidance, cooperation, and immense encouragement. I also extend my thanks to the entire faculty and staff of the Department of Computer Applications, TKMCE, who has encouraged me throughout this work.

I also express my thanks to my loving parents, brother, and friends, for their support and encouragement in the successful completion of this project work.

GANGA KRISHNAN.G

ABSTRACT

Numerous aspects of daily life are still being transformed by technologies and services that are geared toward intelligent transportation systems and smart automobiles. Automatic Number Plate Recognition has ingrained itself in our culture and is here to stay. The approach used to examine a vehicle's license plate in a photo or video collection is referred to as Automatic License Plate Recognition (ALPR) or Automatic Number Plate Recognition (ANPR). Intelligent Transportation Systems are made possible by ANPR technology, which also reduces the need for human interaction. This project aims to find out the best algorithm for license plate detection. The project uses four deep neural networks such as CNN, VGG16, VGG19, and YOLOV3 to detect the license number plate and evaluate the performance of the models in terms of accuracy and find out the best model.

CONTENTS

LIST OF FIGURES

Chapter 1
INTRODUCTION

A technique for reading a vehicle's license plates in a succession of photos or videos is Automated License Plate Recognition, sometimes referred to as ALPR or Automatic Number Plate Recognition (ANPR). Automatic Number Plate Recognition Systems are getting increased attention as a result of their application in traffic management systems that have been installed in a number of nations for tasks including traffic law enforcement and security monitoring. Aside from controlling entry and exit in parking lots, acquiring toll money, and keeping an eye on security in restricted places like military camps and guarded sanctuaries, ANPR systems can also be employed.

These ANPR technologies are frequently used to bolster security in certain locations and stop fraud. They could be useful, for instance, while hunting for automobiles used in theft or other crimes. This activity involves a lot of labour, time, and resources, barring ANPR solutions. Additionally, it is extremely challenging for a person to precisely memorize or read a moving vehicle's license plate, and physical interference in such activities may lead to inaccurate interpretations. If a vehicle is present in the frame of an image or video stream that an ANPR system receives as input, it displays the information on the license plate, often as text. These gadgets contain a camera for taking images of cars. Depending on the system's parameters, the images could be in colour, black & white, or infrared.

In order to properly identify a vehicle, a license plate is "a metal or plastic plate attached to the vehicle", according to the basic definition. However, a machine is unable to comprehend this definition. A definition that a machine can understand is necessary to identify a license plate. A license plate may be characterised by its characteristics as "a rectangular region of a vehicle with a higher density of horizontal and vertical edges." Numerous methods to solve the license plate detection problem have been proposed based on these properties. These algorithms are divided into those that are based on deep learning and those that are based on more conventional computer vision methods. To address the particular difficulties of license plate identification, a number of pre-processing steps are taken before character segmentation and recognition.

Rapid urbanization is a key development in the contemporary environment. People leave village areas and primarily decide to live in a city. As traffic in these places increases, local governments frequently overlook the mobility demands of residents and visitors, both now and in the future. A significant amount of traffic flow analysis using ANPR is being done to support intelligent transportation.

An enormous number of moving cars may be detected and scanned using ANPR technology, which many elements of the modern digital world may include. Although ANPR technology comes in a variety of shapes and sizes, they all serve the same fundamental purpose, which is to offer a very precise way to scan a vehicle without human intervention. ANPR technology can be used for many different services, including entry control, parking management, toll collection, customer billing, order tracking, intelligent traffic, policing and security services, user support and advice, the regulation of red lights and lanes, queue length assessment, and many more.

Number plate recognition with a camera includes gathering pictures of license plates from the target area. By processing still photographs or a photographic video using a variety of image processing-based recognition techniques, it is possible to convert the captured images into an alpha-numeric text. After getting a crisp picture of surrounding area or the target car, any ANPR system should concentrate on how trustworthy its algorithms are. The proper operation of ANPR and other smart car technologies depends on a number of key algorithms.

Character recognition (CR) is a procedure that an ANPR system typically goes through after picture acquisition (the system's input), number plate identification, and image pre-processing as output from the system. Since the license plate detection and recognition stages can be clearly interpreted, many studies have used these stages to gauge performance. Another reason is that the majority of deep learning and machine learning techniques are trained using losses that are specified for each of these steps. The information may be accessed and utilized for any required post-processing operations once the vehicle has been correctly identified.

Depending on the type and resolution of the camera being used, the mounting location, and any lighting or illuminating assistance, and other environment and device constraints, the image captured from the scene may encounter some complexity. The methods used to solve ANPR under the current constraints are computationally complicated and time-consuming. The system has been difficult to segment and recognize the characters due to a number of issues, including the diversity of plate character viewpoint, shape, format, and fluctuating light

conditions at the time of picture collecting. Convolution Neural Networks (CNN) is a technique with excellent performance that has been employed recently. However, the CNN architecture's max-pooling layers are prone to information loss when feature maps are down-sampled.

In many other systems where authorization is essential, such as smart parking system, toll payment processing systems, etc., Automatic Number Plate Recognition (ANPR) also plays an essential role. By automating the procedure, security officials can significantly reduce their workload. Computer vision technology has made significant advancements on a number of practical challenges in recent decades. Previously, car number plates could be recognized using template matching techniques by measuring their width, height, contour, etc. Nowadays, number plate recognition uses numerous deep learning models that have been trained over vast amounts of data.

The primary causes of traffic congestion and violations are the exponentially rising number of vehicles on the road. The Automatic Number Plate Recognition system's goals are to automate traffic control and decrease traffic offenses. Diverse tactics are employed in India, however, they are not particularly successful. The use of machine learning techniques to track vehicles and license plates is already possible. The difficulty of these algorithms' real-time background processing, however, causes them to fail in real-time. Therefore, it is urgently necessary to create an automated system that would assist in tracking the automobiles by accurately tracing their license plates.

The massive need for technology for traffic monitoring and management is sparked by the sharp increase in vehicle traffic on the roads. It is quite impossible to manually follow moving cars on the road in this circumstance. Losses in time and labour will occur. Even if it is operated manually, that will show incredibly tough and incredibly mistaken operations. Four neural networks were used to train the system, and its performance was assessed in order to determine the optimal approach. The suggested approach aims to boost and improve efficiency. Also, to find the best algorithm among different deep learning neural networks to support intelligent transportation systems.

1.1 PROBLEM STATEMENT

Traffic control, vehicle identification, and location monitoring are all manual processes that increase time, expense, and effort. The introduction of ANPR technology helped to solve this issue. The suggested system is designed to increase and enhance efficiency.

1.2 OBJECTIVE

The main goal of the project is to determine the best algorithm for license plate detection in order to develop an automated number plate recognition system that will support intelligent transportation systems by reducing the time, expense, and human effort required for the current number plate recognition process.

Chapter 2
LITERATURE SURVEY

The full analysis of the literature that is relevant to a certain issue is known as a literature review. When doing a literature review, research questions are first developed, after which one attempts to provide a solution by looking up and analyzing pertinent material. The ability to re-analyze the study's findings can lead to the development of fresh ideas, which is one benefit of literature reviews. A literature review summarizes and explains the whole and most recent body of information about a subject that may be found in academic books and journal articles. one sort of literature review is completed as a stand-alone assignment for a course, and the other type is written as part of the introduction to or preparation for bigger work, sometimes a thesis or research report. Both types of literature reviews may be assigned to students at the university level. Depending on the type of review you are writing, your topic, point of view, and type of hypothesis or thesis argument will all be influenced. Reading published literature reviews or the introductory sections of theses and dissertations in your own field of study, analyzing the organization of their arguments, and making notes of how they approach the topics are some ways to appreciate the distinctions between these two types.

2.1 PURPOSE OF THE LITERATURE SURVEY

i. It makes research on a certain issue accessible to readers by picking high-quality papers or studies that are relevant, significant, important, and valid and summarising them into a single report.

ii. It is an ideal starting place for scholars who are just starting out in a new field. by requiring them to describe, assess, and compare original research in that field.

iii. It assures that researchers do not replicate previously completed work.

iv. It might point the way forward for future study or suggest areas to focus on.

v. It emphasizes the most important results.

vi. It finds discrepancies, gaps, and contradictions in the literature.

vii. It gives a constructive appraisal of other scholars' methodology and approaches.

2.2 RELATED WORKS

A system for speeding up car registration in Peru was created by Miluska Valdeos et al. [1], however, it requires accurate extraction and location identification of the license plate. Here, the OpenCV library and the Python programming language are utilized. A neural network that has been trained to locate the location of the license plate is also used by YoloV4 to facilitate the deployment of an optical character recognition system (OCR).

In order to enable the personnel to use the mobile phone to swiftly and precisely gather vehicle information to complete the vehicle inspection, Pengjie Huang et al. [2] suggested an ANPR system. A system built on the Android platform and utilizing the deep neural network framework TensorFlow and the open-source computer vision library OpenCV is created. The system successfully completes the duties of locating the location of the vehicle's license plate as well as segmenting and identifying the characters on the plate. The outcomes of the experiments demonstrate that this technique is somewhat adaptable and has a positive impact on license plate recognition.

For the recognition of license plate characters, Xudong Fan et al. [3] presented a segmentation-free network (CNNG) and a reliable license plate detection network (CA-CenterNet). No matter how the license plates are rotated or distorted, CA-CenterNet can identify not only the center of each plate, but also four vectors pointing to each of its four corners. This ability allows us to correct the distorted license plates in the source photos. The characters in the detected license plates can thus be reliably identified using CNNG without character segmentation.

A plan for managing the cars in parking lots was put forth by Hui Xu et al. [4]. The Flask web framework and the Python programming language were used to create the web-based system. The camera's photographs were downloaded using OpenCV, and Open ALPR was used to find and examine license plates. The system runs on Linux and Raspbian operating systems. The experiments show that the technology can be applied to parking cars and can identify 85% of camera-captured photos.

SegNet, a segmentation technique, is employed in a system created by P Lavanya Kumari et al. [5] to perform LPL. Semantic pixel-by-pixel segmentation is accomplished using Deep Convolution Neural Network (CNN) architecture. Utilizing adaptive contrast enhancement and Gaussian filtering, the input image is pre-processed. The LP region is located

and segmented using a semantic segmentation network (SSN). For segmentation, a deep encoder-decoder network is employed. A classification is given to the segmented LP zones by CNN.

A system that includes imaging, number pad placements, alphanumeric character truncation, and OCR was proposed by Milan Samantaray et al. [6]. The system's ultimate goal is to design and develop effective image processing processes and techniques to place a license label on an image from the Open Computer View Library. Python programming language and the K-NN technique are employed here. The study offers a different method for implementing AVLPR systems using free software, such as Python and the Open Computer Vision Framework (openCV).

In research, Annisa Firasanti et al. [7] compared the effectiveness of Canny Edge and Otsu Thresholding as segmentation algorithms for recognizing license plate edges. Before the license plate text is recognized by the OCR and Tesseract Library techniques, the license plate picture data must go through a number of processing steps. Data processing is done using Raspberry Pi. The two methodologies were put to the test using 30 vehicle samples throughout the course of three different time periods—morning, afternoon, and night time. According to the findings of the studies, Canny Edge performs better since it can identify 100% of the edges, as opposed to Otsu Thresholding, which can only detect 70% of the edges of the total dataset. Generally, the system developed correctly recognizes number plates with an average accuracy of 72%.

In order to detect the data on photos and videos, Tanushri Bhagat et al. [8] studied Automatic License Plate Recognition (ALPR) utilizing a variety of Python modules. The article outlines a brand-new image processing technique that will be applied to the identification of Indian license plates in videos. Additionally, it highlights the necessity of this system in India as well as the significance of high-intensity photographs for their detection. Additionally, some potential future developments that will improve the existing functioning paradigm while maintaining the security and privacy of user data are suggested. The Local Binary Pattern method is used to find the ALPR system, and the HTTP server's Beanstalkd queue is then used to identify it, completing the software implementation.

Chapter 3
METHODOLOGY

The development of an automated number-plate detection system includes various steps- dataset collection, data pre-processing, algorithm selection, training the model, and number plate localization.

3.1 SYSTEM ARCHITECTURE

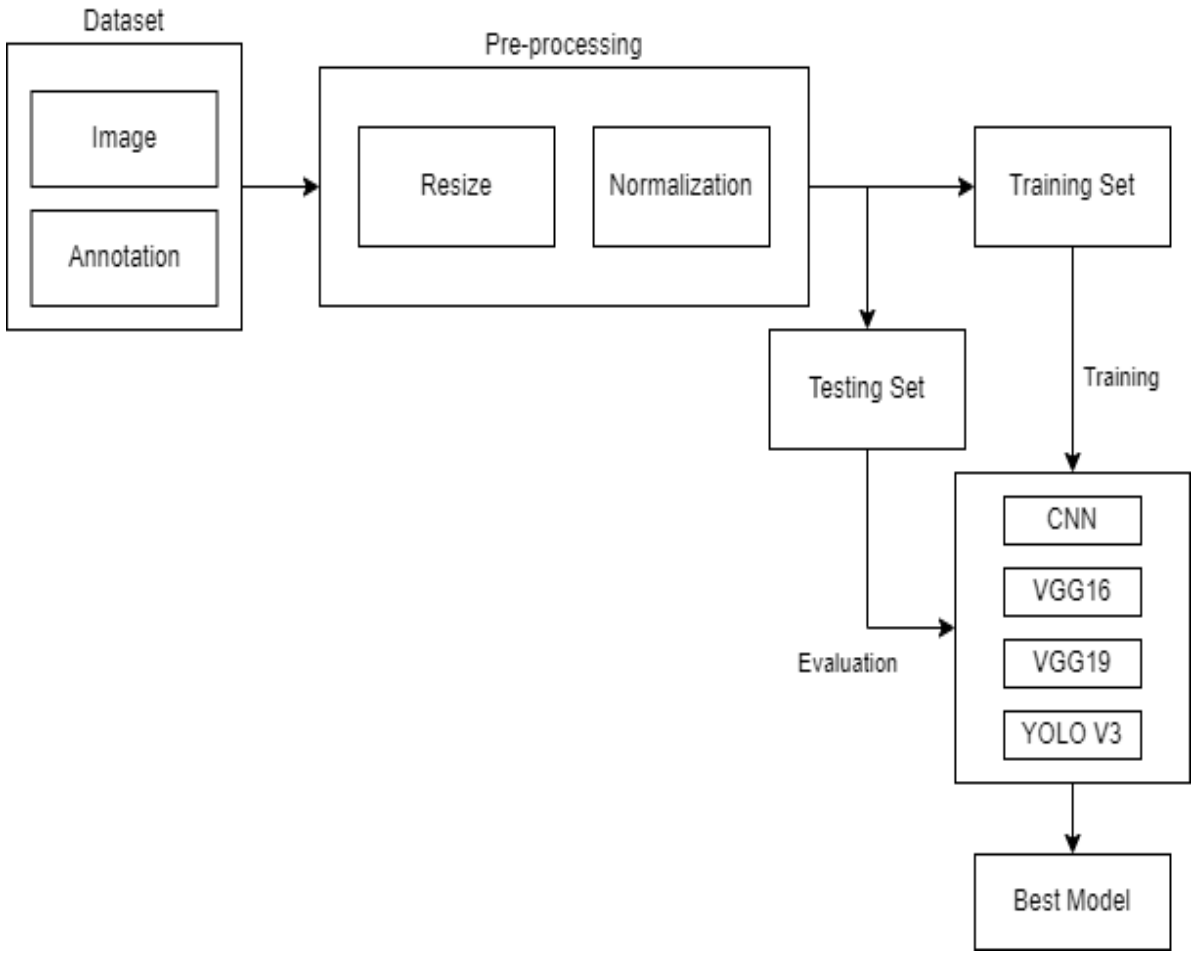

Fig 3.1 Workflow of the System

Steps:

- **Data Collection**

The procedure starts with gathering information. The process of collecting and examining data from different sources is known as data collection. For training, the dataset from Kaggle is used. The dataset contains 2 folders- the image folder and the annotation folder. The image folder contains different car images. It consists of 433 car images with a .png extension. The annotation folder contains annotation files for every image in the image folder. Annotation file contains details of each image file like height, width, depth, name, etc and it is in XML format.

Fig 3.2 Images from Dataset

- **Data Pre-processing**

Data pre-processing is the process of modifying raw data to be utilized by a machine learning model. Pre-processing improves the data so that it can be processed later. Machine learning models cannot be created with real-world data since it frequently contains noise, missing values, and may be in an undesired format. The training procedure is carried out only after the data is pre-processed. Images make up the data, hence picture pre-processing is used. Pre-processing includes operations like picture scaling and normalization.

Resize modifies the dimensions of the photographs and, if desired, scales them. Annotations are proportionally changed. The possible values for each pixel are 0 to 256. A colour code is represented by each digit. The computation of big numeric values could be more difficult when utilizing a Deep Neural Network to process the image as it. The users can also normalize the

data to fall between 0 and 1 to lessen this. The calculations will be simpler and quicker because the numbers will be tiny. With the exception of 0, the range is 255 because pixel values range from 0 to 256. The range will now be from 0 to 1 after dividing all of the values by 255.

- **Train Model**

Training the images in the train folder comes next. It is aware of all the car's attributes. Simply said, training a model entails learning (deciding) appropriate values for each weight and bias from labelled samples. The consequence of a poor prediction is loss. In other words, the loss is a measure of how poorly the model predicted a single case. The loss is zero if the model's forecast is accurate; otherwise, the loss is higher. Finding a set of weights and biases that, on average, have low loss across all examples is the aim of training a model. MSE is the loss function in use here.

The average squared loss across all examples in the collection is called the mean square error (MSE). Divide the total number of cases by the sum of the squared losses for each unique example to obtain MSE.

For training 4 algorithms are used- Convolution Neural Network (CNN), Visual Geometry Group 16 (VGG16), Visual Geometry Group19 (VGG19) and YOLO (You Only Look Once) V3. Train these 4 algorithms separately and evaluate the performance of these algorithm to compare the performance of these algorithms to find the best algorithm for number plate detection.

```
train = model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=10, batch_size=32, verbose=1)
```

Fig 3.3 Training

- **Test Model**

The programmer enters input and monitors the machine's behaviour and logic during machine learning testing. Therefore, the goal of testing is to clarify that the logic the machine learns is consistent. Even after several calls to the application, the logic shouldn't alter. Test the four models CNN, VGG16, VGG19 and YOLOV3 to evaluate the performance of the model.

Fig 3.4 Testing

3.2 ALGORITHMS

For number plate recognition, four algorithms such as Convolution Neural Network (CNN), Visual Geometry Group 16(VGG16), Visual Geometry Group19(VGG19) and YOLO (You Only Look Once) V3 are used.

3.1.1 CONVOLUTIONAL NEURAL NETWORK (CNN)

The detection of the distracted driving problem is solved using the Convolutional Neural Network (CNN) model. CNN has demonstrated to be incredibly effective in classifying pictures, and as a result, is an excellent fit for this issue. It is important to remember that CNNs frequently experience overfitting, which happens when a model adapts too well to trained data but performs badly on untrained data and is referred to as undergeneralizing. It is best to reduce this problem.

Instead of having a fully connected layer for every pixel, CNNs only contain the weights necessary to examine a small area of the image at a time. Convolution layers, pooling,

and activation functions are frequently used in the process, however not always in that same sequence. The original image can have each of these three distinct procedures applied as a separate layer, usually numerous times. To identify a picture appropriately, a fully linked layer (or more layers) is added at the end. Finding the precise combination that provides the best performance might be challenging due to a large amount of highly variable combinations and permutations. The community and research, who have luckily produced some promising findings and made their CNNs openly accessible for usage and improvement, accelerate the development of CNN designs.

Three main types of layers make up the CNN: fully connected (FC), pooling, and convolutional layers. The result of stacking these layers is a CNN architecture.

Activation function

ReLU and Sigmoid activation functions are mostly employed in this study.

ReLU stands for Rectified Linear Unit. The ReLU is now the activation function that is used most frequently worldwide. mostly because convolutional neural networks or deep learning use it. The ReLU for short, can have two types of outputs either the input or zero , if the input is positive or negative. The derivative and the function are both monotonic. The issue is that because every negative number instantaneously becomes zero, the model is less able to fit or train from the data. Because of the improper mapping of the negative values, every negative input to the ReLU activation function ultimately ends in a graph value of zero.

Range will be from zero to infinity

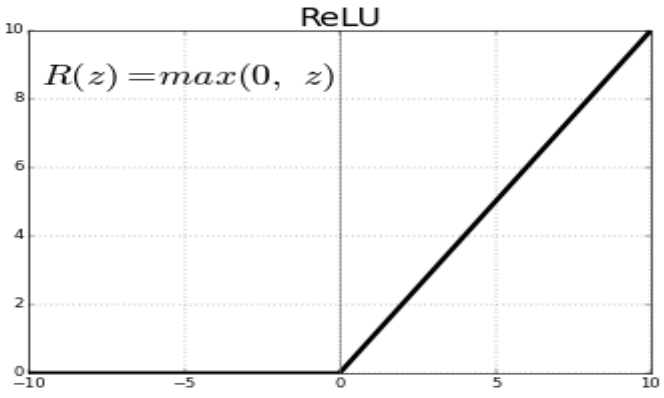

Fig 3.5 ReLU Activation function

The Sigmoid Function curve seems to be in the shape of a S. Sigmoid function is primarily utilized since it occurs between (0 to 1) It's used especially for models whose output is a probability prediction as a consequence. The ideal choice is the sigmoid since everything has a probability that only ranges from 0 to 1. The function might take many different shapes. As a result, the users can calculate the slope of the sigmoid curve between any two points. The derivative is not monotonic, despite the function being monotonic. A neural network may become stuck during training as a result of the logistic sigmoid function. For multiclass classification, the softmax function, a more extended logistic activation function, is utilized.

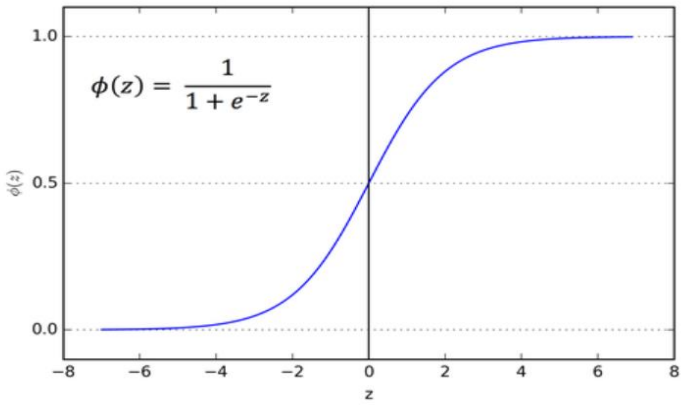

Fig 3.6 Sigmoid Activation function

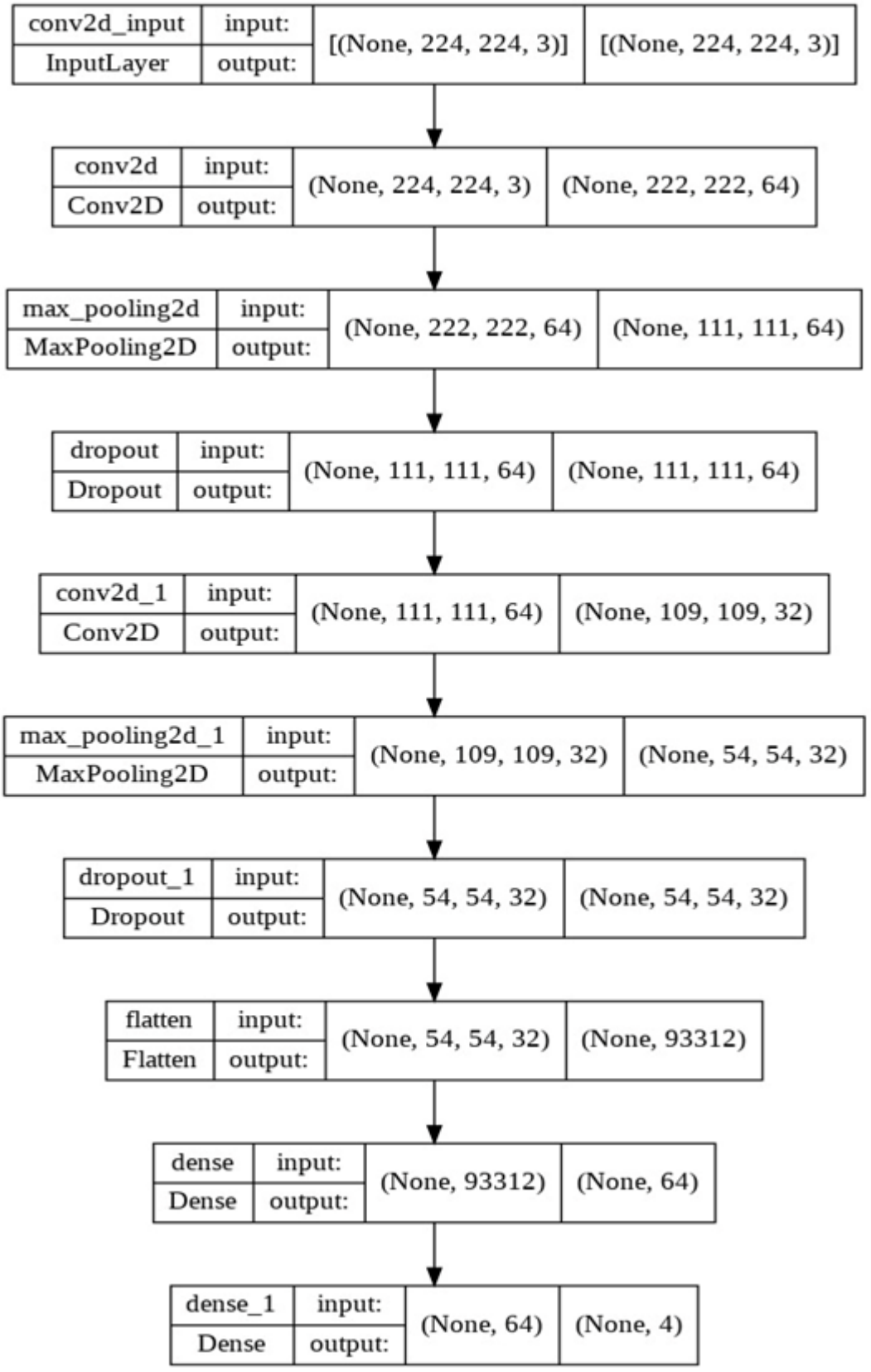

Fig 3.7 CNN with dimensions of Input and Output

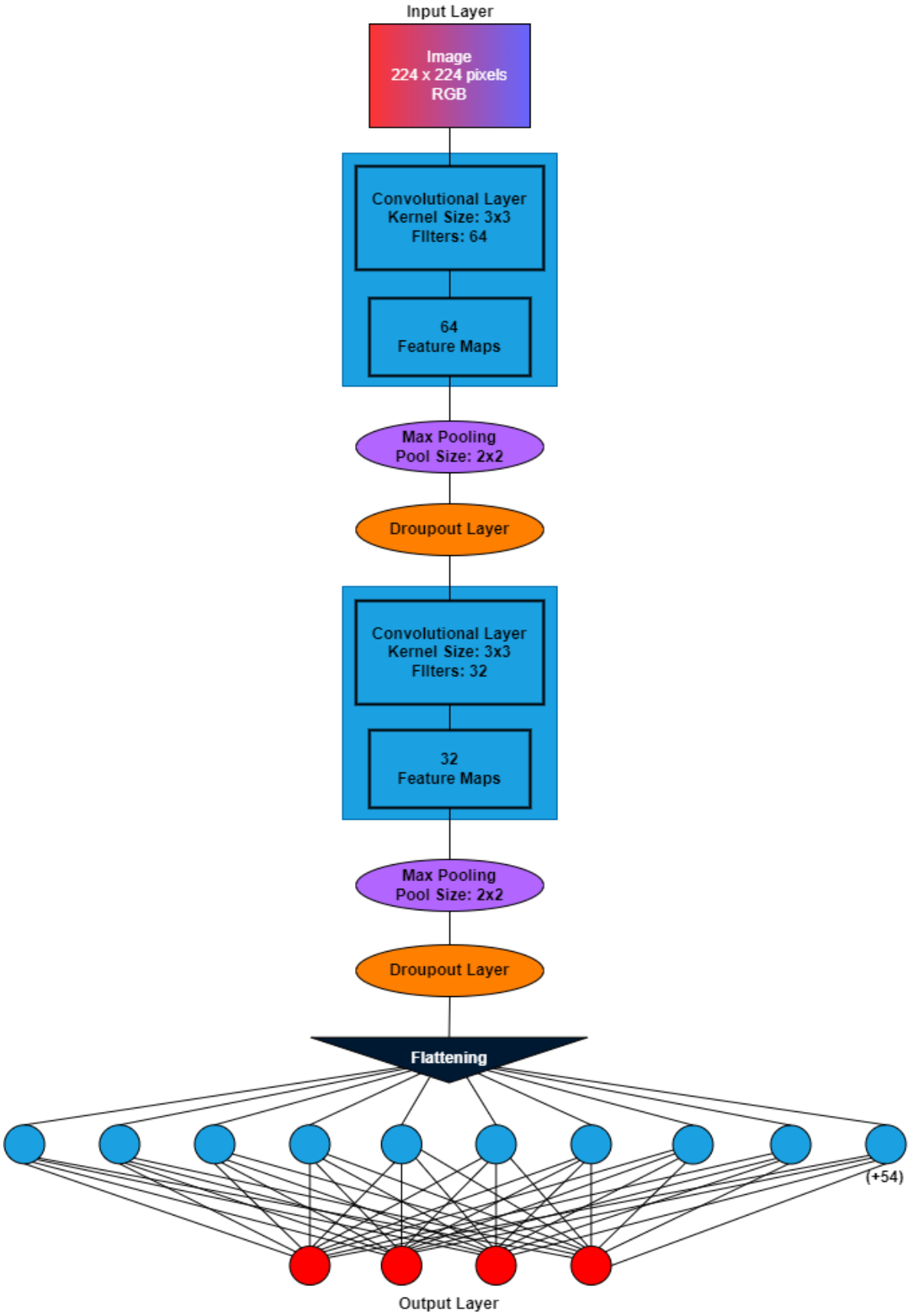

Fig 3.8 Visualization of CNN

STEPS:

- A convolutional layer with 64 filters and a 3,3 kernel makes up the input layer. The input shape is (224,224,3) and the activation used is ‘ReLU’.
- Next, the Max-Pooling layer have a pool size (2,2).
- In order to prevent an overfitting issue, it is then linked to a dropout layer with a dropout rate of 0.1.
- Next, the output is given to a 2D Convolutional layer with 32 neurons and ‘ReLU’ activation.
- The output is given to a Max-pooling layer and then it is connected to Dropout layer.
- To flatten the feature matrix to a vector for the next Dense layer, a Flatten layer is used.
- Finally, it is connected to a Dense layer, which is the output layer, where the flattened output is supplied to the Dense layer with 64 neurons and "ReLU" activation.
- It maps the feature values to the corresponding output based on the probability measure. In this layer the activation function used is the ‘Sigmoid activation’.

3.1.2 VGG16

VGG16 is an object recognition and classification algorithm that can classify 1000 pictures into 1000 separate classes with an accuracy rating of 92.7%. It is a well-liked technique for classifying photos and is simple to apply to transfer learning.

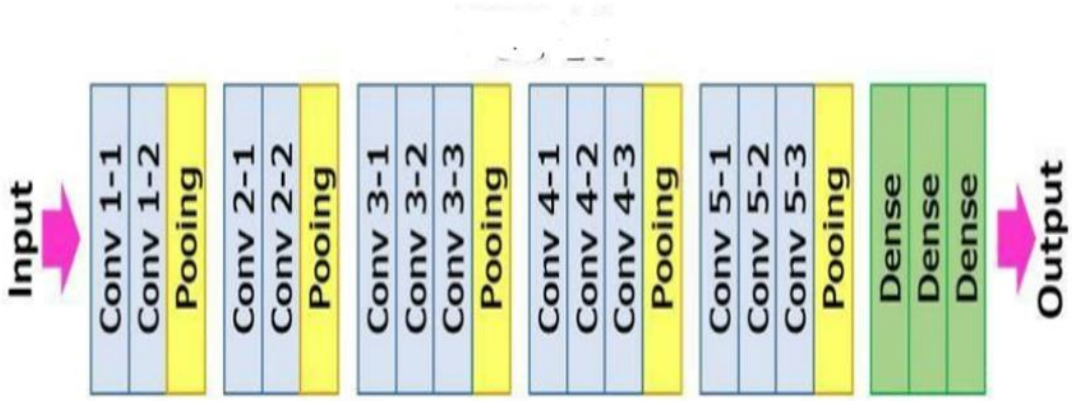

Fig 3.9 VGG16

- 16 weighted layers are represented by the number 16 in VGG16. VGG16 has 21 layers altogether: 13 convolutional layers, 5 Max Pooling layers, 3 Dense layers, although only sixteen of them are weight layers, also known as learnable parameters layers.
- The VGG16 input tensor has a size of 224, 244 and three RGB channels.
- The unique aspect of VGG16 is it consistently employs the same padding and max pool layer of a 2x2 filter with stride 2 and concentrates on having convolution layers of a 3x3 filter with stride 1, rather than having several hyper-parameters.
- The max pool and convolution layers are positioned the same way across the whole design. The Conv-1 Layer consists of 64 filters, the Conv-2 Layer of 128 filters, the Conv-3 Layer of 256 filters, and the Conv-4 and Conv-5 Layers of 512 filters.
- Three Fully Connected (FC) layers surround a stack of convolutional layers; the first two of these layers each have a total of 4096 channels, while the third layer uses 1000 channels to complete a 1000-way ILSVRC classification.
- The final layer is the soft-max layer.

3.1 3 VGG19

It's a variant of the VGG model. 19 weighted layers are indicated by the number 19 in VGG19. In addition to the 16 convolutional layers, it includes3 FC layers, 5 max-pooling layers, and 1 softmax layer.

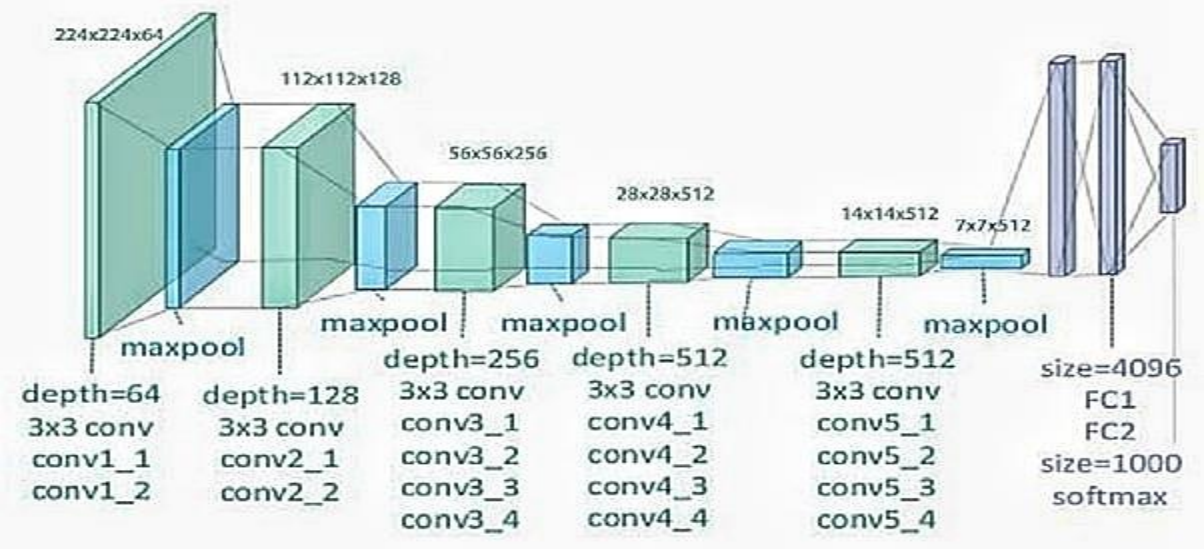

Fig 3.10 VGG19

• This network received a fixed-size RGB picture as input, indicating that the matrix was shaped.

• The only pre-processing that was carried out was to remove from each pixel the mean RGB value calculated for the whole training set.

• They covered the entirety of the image by using kernels that were (3 * 3) in size with a stride size of 1 pixel.

• Spatial padding was utilized to maintain the image's spatial resolution.

• Max pooling was performed throughout a 2 by 2 pixel frame using stride 2.

• TanH or sigmoid functions were utilized in earlier models, thus Rectified linear unit (ReLU) was used to incorporate non-linearity to improve the model's classification and computation time. This proved to be much superior than the other activation functions.

Convolution layer: A convolutional neural network's crucial component is the convolutional layer, which grants the network its name. This layer handles the "convolution" process. As with a conventional neural network, a convolution is a linear approach that involves multiplying a set of weights with input. The multiplication is done between an array of input and a filter, because the procedure was designed for two-dimensional input.

Maxpooling layer: The fact that convolutional layer feature maps preserve the exact location of input features is one of its limitations. This implies that little changes in the feature's location in the input picture will produce a different feature map. Re-cropping, rotation, shifting, and other small alterations to the supplied picture might cause this. Down sampling is a popular strategy used in signal processing to solve this issue. In this case, a reduced resolution version of the input signal is produced, retaining the main structural features but excluding the fine detail that might not be as helpful. Convolutional layers allow for down sampling by adjusting the convolution's stride across the image. Using a pooling layer is a more reliable and typical strategy. After the convolutional layer, a new layer called a pooling layer is introduced. specifically, following the application of a nonlinearity (such as ReLU) to the feature maps produced by a convolutional layer.

Batch Normalization: The network's layers can learn more independently due to the layer of batch normalization. Using it, the output from the prior layers is normalized. As part of the normalising process, the input layer is scaled by activations. Batch normalisation improves

learning efficiency and may be applied as regularisation to avoid model overfitting. The layer is connected to the sequential model in order to normalise either the input or output. It might be implemented anywhere between the levels of the model. After describing the sequential model and just after the convolution and pooling layers, it is typically placed.

Dropout layer: The Dropout layer acts as a mask, eliminating some neurons' contributions to the subsequent layer while maintaining the functionality of all other neurons. It can apply a Dropout layer to the input vector, in which case part of its characteristics are eliminated; alternatively, it can be applied to a hidden layer, in which case some hidden neurons are eliminated. Because they avoid overfitting on the training data, dropout layers are crucial in the training of CNNs.

Flatten layer: A single, extensive continuous linear vector is created by flattening the resulting 2-Dimensional arrays from pooled feature maps.

Average pooling layer: The average for each patch of the feature map is determined using the average pooling approach. In other words, each 2 by 2 square of the feature map's average value is down sampled.

Setting up the network layers and settings is necessary before training begins. The base model is initially VGG19, and the remaining fully linked layers are joined back-to-back. Following VGG19, the Global Average Pooling Layer is introduced. Then, 512 units and the "ReLU" activation function are added to a dense layer. Once more, a thick layer is followed by a batch normalising layer. A dense layer with 256 units and the "ReLU" activation function is attached to it. The network's output layer is then connected to a batch normalisation layer, after which 2 dense layers are added. In contrast to the final dense layer's 2 units and "softmax" activation, the initial dense layer contains 128 units and "ReLU" activation.

The training set is applied to the input layer of the network for training once the network layers and its parameters have been set up. A trained model is produced when training is finished.

3.1.4 YOLOV3

A real-time object identification system called YOLOv3 (You Only Look Once, Version 3) recognises particular things in films, live feeds, or still photos. To find an item, the YOLO machine learning system leverages characteristics that a deep convolutional neural network has learnt. The YOLO machine learning algorithm has three versions, with the third

version being a more accurate version of the first ML method. Versions 1-3 of YOLO were developed by Joseph Redmon and Ali Farhadi.

Version 1 of YOLO was made in 2016, while version 3, which is the one that is heavily addressed in this article, was made in 2018. An enhanced version of YOLO and YOLOv2, YOLOv3 is available. The Keras or OpenCV deep learning libraries are used to implement YOLO. Artificial intelligence (AI) applications employ object categorization systems to identify certain objects in a class as interesting things. The algorithms categorise things in pictures into groups, placing those with similar traits together while ignoring others until specifically instructed to do so.

The convolutional layers' learnt characteristics are handed on to a classifier, as is customary for object detectors, which predicts the detection. In YOLO, a convolutional layer with 11 convolutions is utilized to generate the prediction. The "you only look once" (YOLO) approach predicts using 11 convolutions, and the size of the prediction map is identical to the size of the feature map that came before it.

A Convolutional Neural Network (CNN) called YOLO is capable of swiftly recognizing things. The classifier-based systems known as CNNs may analyse incoming images as organised arrays of data and identify patterns in those patterns (view image below). YOLO has the advantage of being faster than other networks while yet keeping accuracy. The model's predictions are impacted by the whole context of the picture since it enables the model to observe the full image during testing. Methods using convolutional neural networks, such as YOLO, "rank" regions based on how much they match predefined classifications.

High-scoring areas are reported as positive detections of the class to which they correspond most closely. According on which parts of the video score highly in comparison to predetermined classes of cars, YOLO may be utilized, for instance, in a live traffic stream to identify various types of automobiles. Initially, the YOLO V3 algorithm divides a picture into a grid. Each grid cell foretells the presence of a specific number of boundary boxes (also known as anchor boxes) around items that perform well in the aforementioned predetermined classifications.

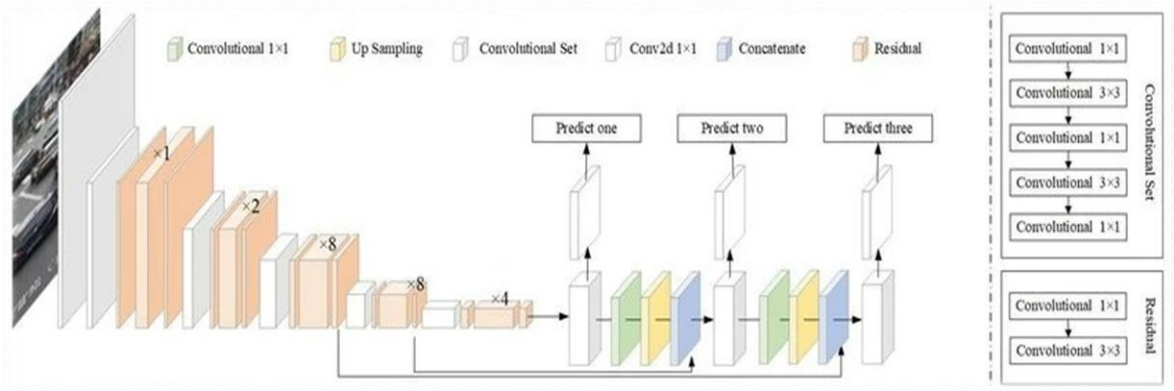

Fig 3.11 YOLOV3

Only one item is detected by each border box, which has a corresponding confidence score indicating how correct it expects that prediction to be. The ground truth boxes' dimensions from the original dataset are clustered to identify the most typical sizes and shapes before being used to create the border boxes. YOLO has been trained to simultaneously do classification and bounding box regression.

3.3 TECHNOLOGIES USED

3.3.1 PYTHON 3

Guido van Rossum is credited with creating the Python programming language. Version 1.0 came out in 1994 after the original version was published in the alt. sources newsgroup in 1991. The 2.x series of versions predominated the market since the release of Python 2.0 in 2000 until December 2008.

Version 3.0, which the development team later produced, was not compatible with the 2.x versions. but did incorporate a few small but substantial modifications. Python 2 and Python 3 are fairly similar to one another, and Python 2 has received backports of certain Python 3 features. But generally speaking, they continue to be incompatible.

Python 2 and Python 3 have both been kept up to date on a regular basis with new releases for each. However, Python 2 will no longer be supported as of January 1, 2020, which is the official End Of Life date. It is advised that users concentrate on Python 3.

Python is a well-liked programming language. Python has grown in acceptance over the last few years. Python was recognized as the year's most popular and desired technology in

the 2018 Stack Overflow Developer Survey, where it came in seventh place overall. Top-tier software development companies utilize Python on a regular basis all across the world. Python is the most widely used programming language in the world, according to data by Dice, and it's also one of the trendiest talents to have, per the Popularity of Index of Programming Languages. Python's popularity and success as a programming language make its developers recognized globally and well-paid.

Since compiled programmes are written in the processor's native language, they often execute quicker than interpreted ones. This could be limiting for some computing-intensive tasks, such graphics processing or challenging arithmetic calculations. In reality, however, the majority of programme execution speed changes are measured in milliseconds, or at most, seconds, and are barely perceptible to a person. For the majority of applications, speed is frequently helpful.

Installing, using, and disseminating the Python interpreter is entirely free, even for commercial reasons, according to the license as open source by OSI that it was created under. Almost every platform that exists has a version of the interpreter, including all variations of Unix, Windows, macOS, cell phones, tablets, and almost everything else you can think of. Even for the remaining 12 users of OS/2, there is a version.

Python has portability. Python code performs line by line interpretation rather than compiled as a whole into native machine instructions, thus it runs on every platform that installed a python interpreter. (This applies to all interpreted languages; Python is not an exception.)

Python is Easy. Python is relatively uncomplicated compared to other programming languages, and its designers purposefully left this way. The number of keywords or reserved words in a language can be used to estimate its complexity. These phrases correspond to certain built-in features of the language and are retained by the compiler or interpreter for particular meaning.

Python 3 features 33 keywords, compared to 31 for Python 2. Comparatively, Visual Basic has over 120, C++ has 62, Java has 53, and there are over 120 in Java, but these later instances likely vary a little depending on implementation or dialect. The structure of Python code is clear and straightforward, making it simple to learn and read. As users can see, the language specification really enforces a readable code structure.

3.3.2 LIBRARIES

Python libraries are reusable pieces of code that the users may want to incorporate into their projects or programmes. Python libraries are not context-specific, unlike libraries for languages like C++ or C. Here, a collection of essential modules is roughly referred to as a "library." Therefore, a library is just a collection of modules. A package management, such as rubygems or npm, can be used to install a library.

- **Python Standard Library**

The precise syntax, tokens, and semantics of Python are collected in the Python Standard Library. The typical Python distribution is included in it. It is constructed in C and controls essential modules such as I/O. All of these characteristics work together to make Python the language as now More than 200 core modules make to the standard library's core. However, in addition to this library, users may also explore a vast collection of many thousands of components via the Python Package Index (PyPI).

1. Keras

TensorFlow serves as the foundation for the Keras algorithm. It was designed to make short experimenting easier. The secret to conducting effective research is being able to move quickly from conception to conclusion.

- Simple: With Keras, the cognitive load on developers is decreased, allowing you to concentrate on the crucial aspects of the issue.
- Strong: Keras offers scalability and industry-leading performance; NASA, YouTube, and Waymo are just a few of the enterprises that employ it.

2. Tensorflow

The open-source TensorFlow framework was developed by the Google Brain team for massive machine learning and numerical computing. In order to make a variety of deep learning (also known as neural networking) models and methods helpful, TensorFlow groups them together. The framework's front-end Python API makes it easy to create apps with it, while high-performance C++ is used to run such programmes.

With the help of TensorFlow, deep neural networks can be trained and used for a variety of tasks. These networks include word embeddings, RNN, sequence-to-sequence

models for machine translation, NLP, and partial differential equation simulations. The best part is that TensorFlow uses the same models that were used for training to provide production prediction at scale.

Developers may use TensorFlow to generate dataflow graphs, which are structures that show how data flows through a graph, or a collection of processing nodes. Every edge connecting nodes in the network, or tensor, is a multidimensional data array, whereas each node itself represents a mathematical operation. All of information is made available to programmers by TensorFlow using the Python language. It provides good ways to define how high-level abstractions may be linked together and is straightforward to understand and use. Python is used to build TensorFlow applications, and Python objects serve as both the nodes and the tensors.

But Python isn't used to carry out the math operations. The transformation libraries that TensorFlow offers are created as high-performance C++ binaries. Python only offers high-level programming abstractions to connect the parts and direct traffic between them.

TensorFlow applications can be run on the majority of real-world targets. However, practically any device can be configured to employ TensorFlow models, which may be used to predict the future.

The abstraction that TensorFlow provides is by far the most important advantage for machine learning development. The programmer may concentrate on the general logic of the programme rather than worrying about the mechanics of implementing techniques or trying to figure out how to link the output of one operation to the entry of another. TensorFlow manages the background information.

For programmers that must troubleshoot and understand TensorFlow programmes, TensorFlow provides extra conveniences. Instead of creating and evaluating the whole graph as a single opaque object, user may assess and modify each graph action individually and publicly using the eager execution mode. The TensorBoard visualization software lets to characterize and examine the behavior of graphs using an interactive, web-based dashboard.

Additionally, TensorFlow gains a lot from Google's assistance, a top-tier business entity. In addition to accelerating the project's rapid development, Google has

developed a number of major TensorFlow-related products and services that make it simpler to deploy and use for increased performance in Google's cloud, the TPU silicon; In-browser and mobile-friendly versions of the framework, as well as many other features, are available online.

It can be challenging to get completely predictable model-training outcomes for some training assignments due to implementation peculiarities in TensorFlow. A model trained on one system occasionally differs somewhat from a model trained on another, even when given the exact same data. The causes of it is unclear (e.g., the location and method of seeding random integers, or specific non-deterministic GPU behaviour). Nevertheless, these problems may be overcome, and the TensorFlow team is looking at additional controls to influence the determinism of a process.

3. Matplotlib

Matplotlib is a package for numerical charting that aids in data analysis. Matplotlib is an efficient Python visualization library for 2D array presentations. To handle the bigger SciPy stack, a multi-platform data visualisation toolkit called Matplotlib was developed and is based on NumPy arrays. John Hunter introduced it for the first time in 2002.

One of the visualization's main benefits is that it allows us visible access to enormous amounts of data in easily understandable ways. In Matplotlib, a number of plot kinds are provided. Plots aid in recognizing trends, patterns, and relationships. They are frequently tools for reasoning about quantitative data.

4. Pandas

The most well-known open-source Python library for data science, data analysis, and machine learning activities goes by the name of Pandas. It is constructed on top of Numpy, a different package that supports multi-dimensional arrays. It offers quick, adaptable, and expressive data structures that make it simple and natural to deal with "relational" or "labelled" data. It aims to operate as the fundamental building block for employing Python for practical data analysis. Data science needs Pandas to function. It offers quick, expressive, and adaptable data structures to make working with structured (tabular, multidimensional, sometimes heterogeneous) and time-series data simple (and intuitive).

5. NumPy

A general-purpose toolkit for handling arrays is called NumPy. It offers a multidimensional array object with outstanding speed as well as capabilities for interacting with these arrays. The foundational Python module for scientific computing, to put it simply. It offers basic scientific computer capabilities as well as sophisticated math features.

Additionally, it provides functions for working with matrices, the Fourier transform, and the area of linear algebra. NumPy was created by Travis Oliphant in the year 2005 There are no restrictions on how the project may be utilized because it is entirely accessible. NumPy is the acronym for Python's numerical language. The equivalent of arrays in Python are lists, although they take a long time to execute. NumPy seeks to provide array objects that are up to 50 times faster than standard Python lists.

The NumPy array object is referred to as ndarray, and its numerous auxiliary methods make using ndarray very easy. In data analysis, where effectiveness and resource management are crucial, arrays are widely utilized. In contrast to lists, NumPy arrays are kept in a single continuous location in memory, making it very easy for programs to access and modify them. In computer science, this characteristic is known as the locality of reference. This is the primary factor that makes NumPy quicker than lists. Additionally, it is made to work with the newest CPU architectures.

6. OpenCV

A Python open-source computer vision library called OpenCV is used in applications like artificial intelligence, machine learning, facial recognition, etc. The term "computer vision" (abbreviated as "CV") in OpenCV refers to a branch of research that enables computers to comprehend the content of digital images like pictures and movies. To comprehend the substance of the images is the goal of computer vision. It takes the description of the images—which may be of an item, a written description, a three-dimensional model, etc.—and extracts it from the images.

The openCV is used for computer vision due to the following reasons:

- OpenCV may be downloaded for free.
- The OpenCV library is quite quick because it is written in C/C++. Python may now be used with it.
- It might use as little as 60–70 MB of RAM.
- Computer vision is portable and may be utilized on any C-capable device due to OpenCV.

7. Seaborn

For creating statistical visualisations, Python's Seaborn visualisation module is amazing. It provides wonderful default colour schemes and styles to improve the aesthetic appeal of statistics charts. The Pandas data structures are closely tied to it and it is based upon the Matplotlib toolbox. With Seaborn, visualization will be at the heart of data exploration and comprehension. For a better comprehension of the dataset, Specifically, it provides dataset-oriented Interfaces that let us shift between various visualizations of the same variables.

8. Scikit

A free machine learning toolkit for Python is called Scikit-learn. It supports Python's NumPy and SciPy libraries, as well as a number of methods including support vector machines, random forests, and k-neighbours.

David Cournapeau created it in 2007 as part of a Google Summer of Code project under the name scikits.learn. Later, in 2010, FIRCA's took this project to a new level and released the first beta version (v0.1) on February 1st.

3.3.3 VS CODE – IDE

In Visual Studio Code, flexible source code editors are combined with strong development tools including debugging and IntelliSense code completion. Due to its support for hundreds of languages and features like syntax highlighting, parenthesis matching, and more, Visual Studio Code (VS Code) is easy to use and productive. Due to simple modification, simple shortcut keys, and other factors, users may explore the code effortlessly.

Tools having more code knowledge than just blocks of text are typically helpful for serious coding. The capabilities which also come included with Visual Studio Code include code refactoring, deep semantic code comprehension and navigation, and IntelliSense code completion. To perform regular chores and speed up everyday operations, VS Code also integrates with build and scripting tools.

Chapter 4
RESULT AND DISCUSSION

The system evaluates the performance of four deep neural networks. Finding the best object detection method is the main objective of this study. i.e., a best object detection algorithm can accurately find out the license plate region so that the license plate number can be extracted successfully.

The system automatically detects the license plate using four deep learning algorithms. The input image to the system is a car image and detects the number plate using the deep neural networks CNN, VGG16, VGG19, and YOLOV3 separately. Finally, evaluate the performance of the four deep neural networks in terms of accuracy to find out the best algorithm for license plate detection.

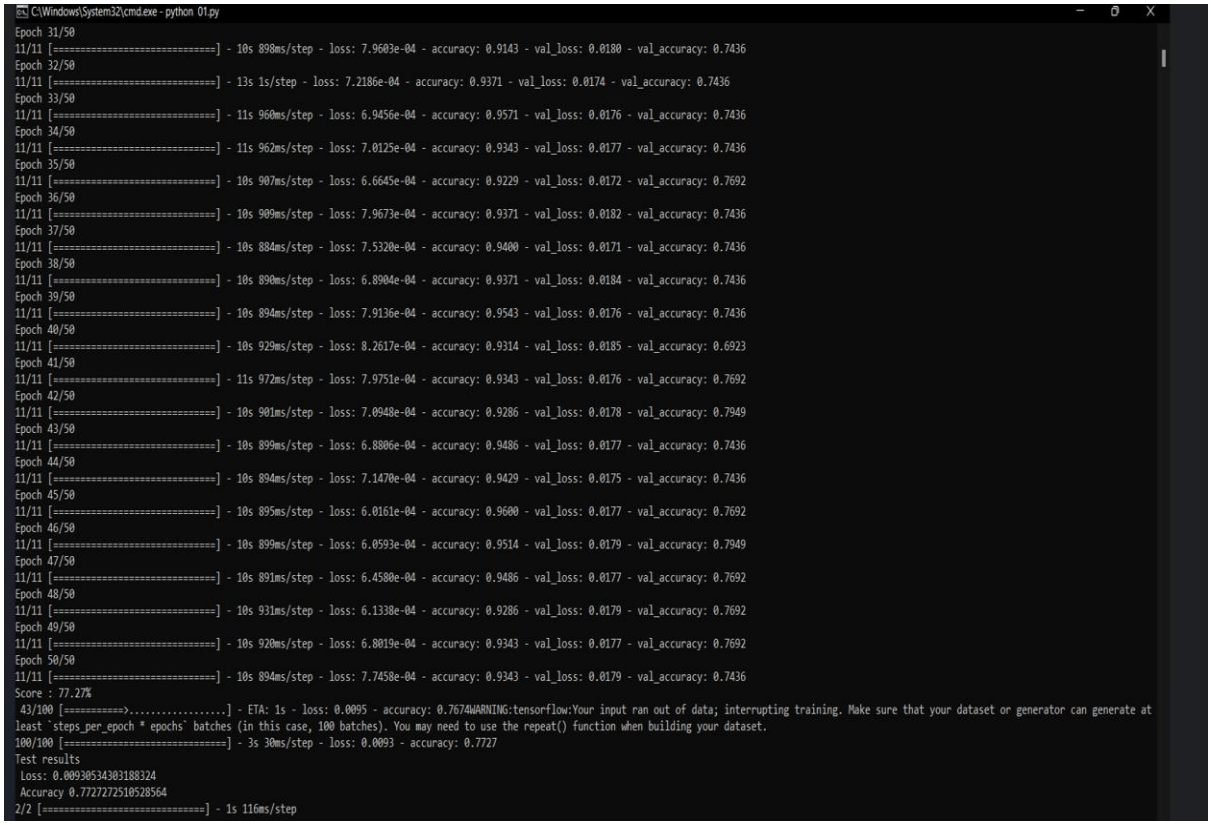

Fig 4.1 Test Results of CNN

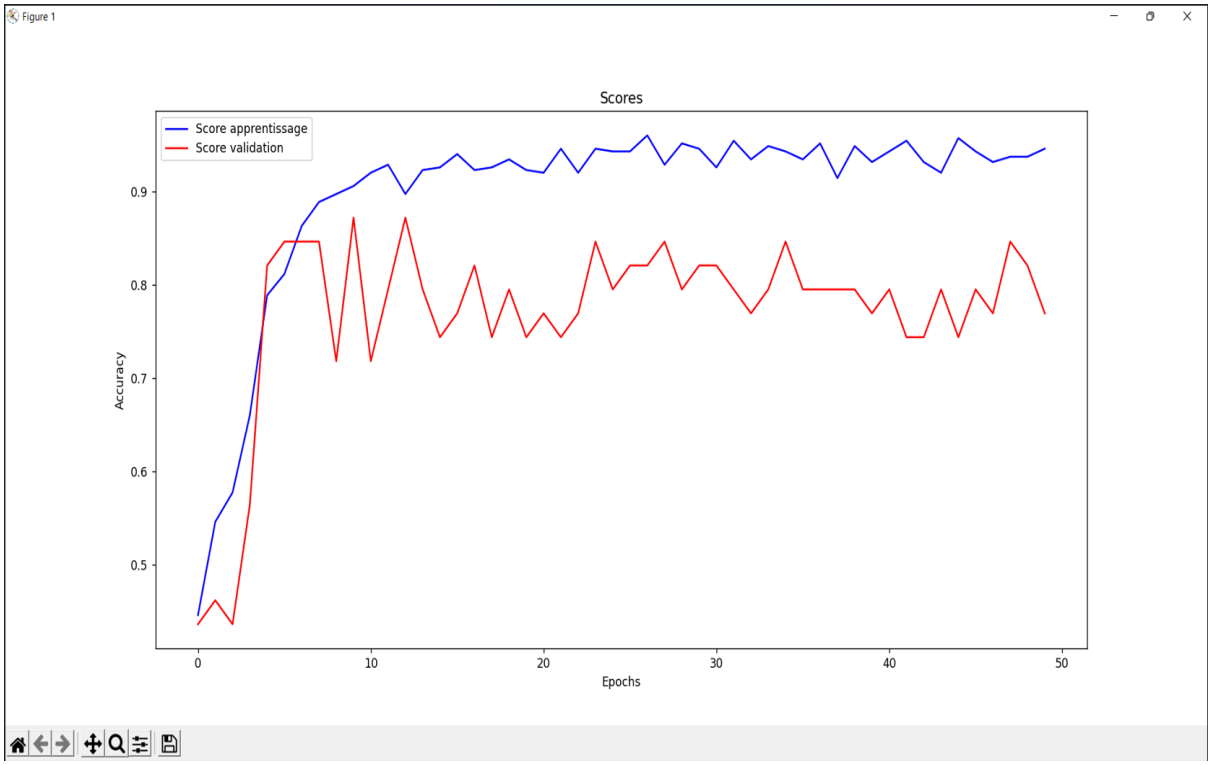

Fig 4.2 Epoch V/S Accuracy Graph of CNN

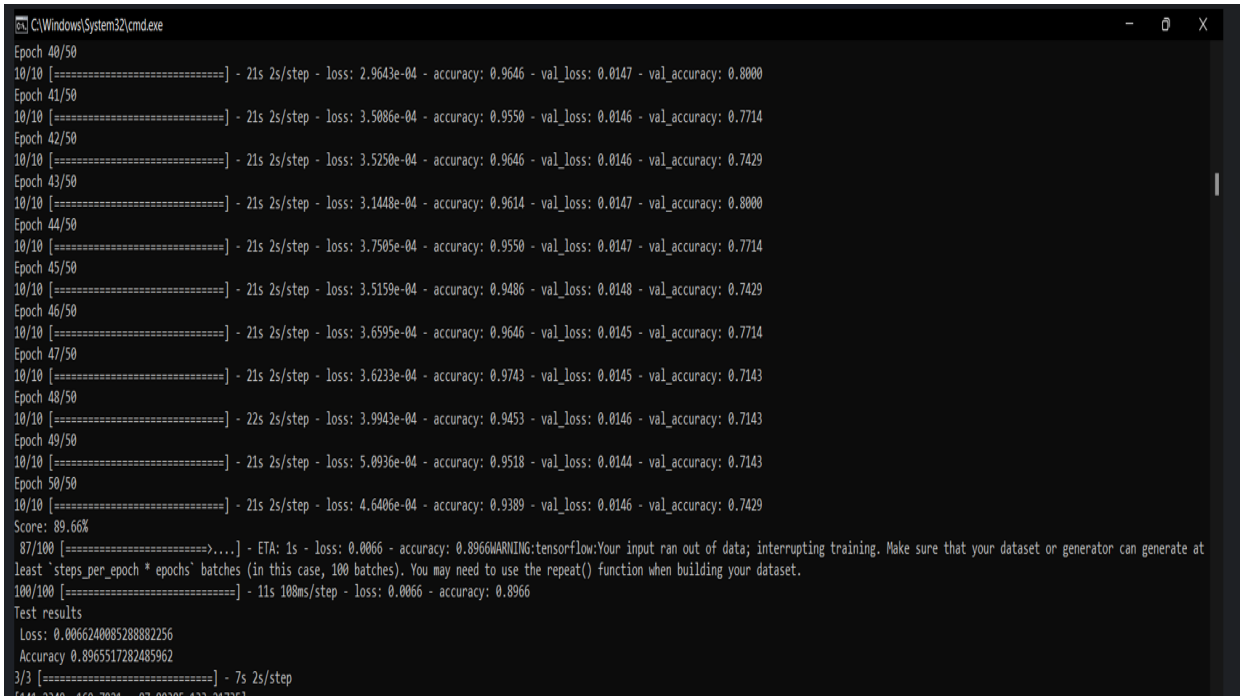

Fig 4.3 Test Results of VGG16

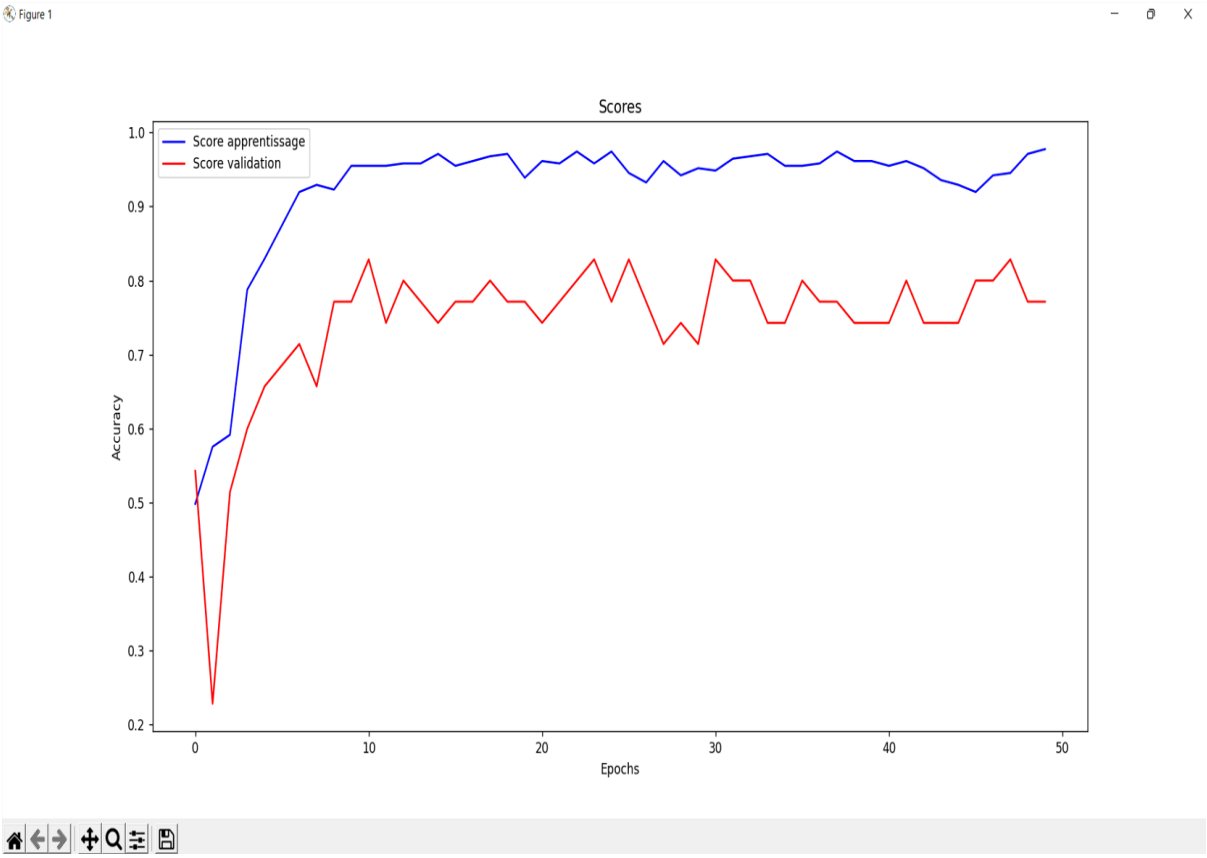

Fig 4.4 Epoch V/S Accuracy Graph of VGG16

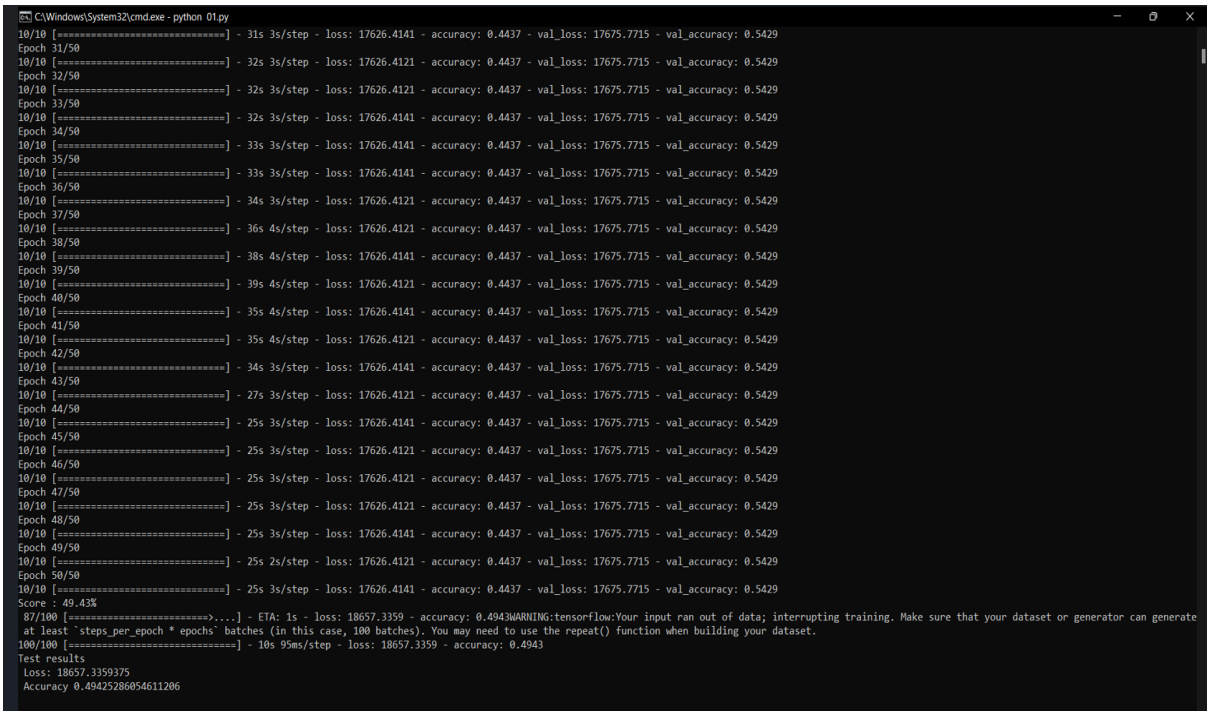

Fig 4.5 Test Results of VGG19

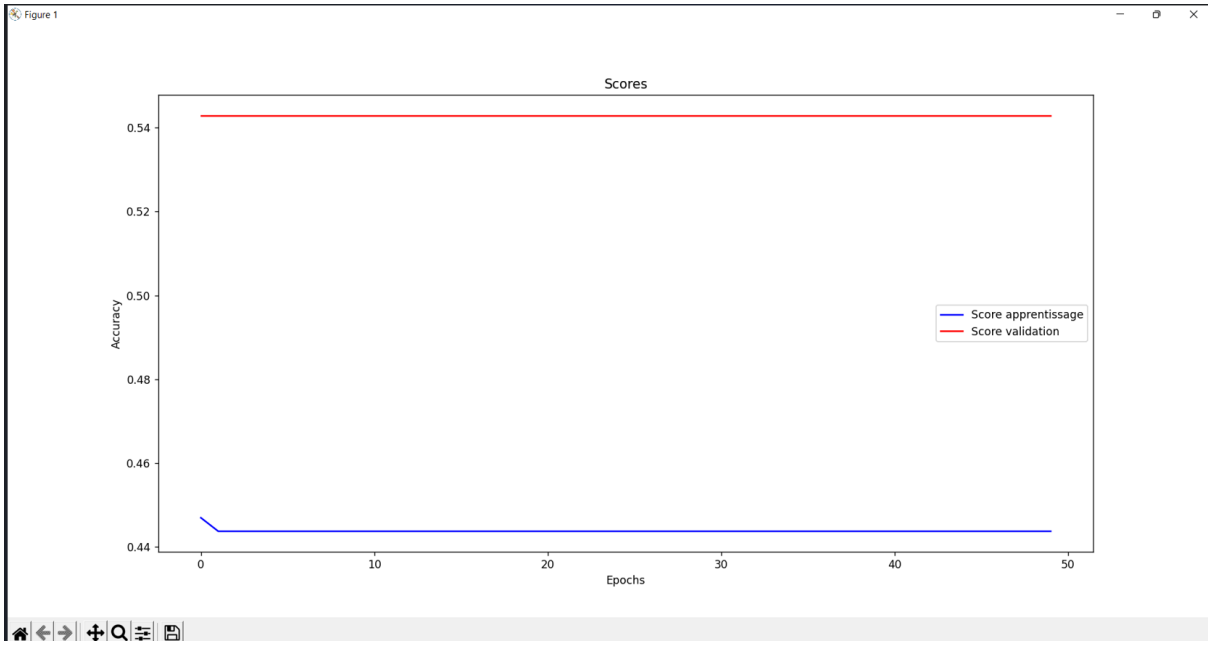

Fig 4.6 Epoch V/S Accuracy Graph of VGG19

Comparison

The VGG16 achieves the highest performance while evaluating the performance of each algorithm using the test set. VGG19 exhibits the least accuracy as well. The VGG16 is the final model employed for number plate recognition. CNN shows an accuracy of 77%, VGG16 shows an accuracy of 89%, VGG19 shows an accuracy of 49%, and YOLOV3 shows an accuracy of 78%. Among these, VGG16 exhibits precise license plate recognition.

Chapter 5
CONCLUSION

The study compares four deep neural networks in terms of accuracy to find out the best model for the ANPR system. For the study, dataset with both image and annotation files are used for training. The dataset is collected from the public repository for the study. The pre-processing module receives the dataset before training and prepares the data. For training the model, CNN, VGG16, VGG19, and YOLO V3 algorithms are used. Train these algorithms separately on the pre-processed dataset and perform an evaluation on these models. The evaluation result shows that VGG16 performs well on this dataset with an accuracy of 89.6% i.e., VGG16 outperforms all other algorithms for number plate detection.

5.1 FUTURE ENHANCEMENT

Currently, the system shows that VGG16 achieves the highest accuracy. But it fluctuates according to the brightness of the light. VGG16 shows lowest accuracy in poor lighting conditions as compared to YOLOV3. So, developing a system that switches algorithms automatically between yolov3 and vgg16 according to the light intensity would result in a more efficient ALPR system.

Chapter 6

REFERENCES

[1] M. Valdeos, A. S. Vadillo Velazco, M. G. Perez Paredes, and R. M. Arias Velasquez, "Methodology for an automatic license plate recognition system using Convolutional Neural Networks for a Peruvian case study," *IEEE Lat. Am. Trans., vol. 20, no. 6, pp. 1032–1039*, 2022.

[2] P. Huang and W. Wang, "Research and design of automatic license plate recognition system based on android platform," in *2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC)*, 2022.

[3] X. Fan and W. Zhao, "Improving robustness of license plates automatic recognition in natural scenes," *IEEE Trans. Intell. Transp. Syst., pp. 1–10*, 2022.

[4] H. Xu, X.-D. Zhou, Z. Li, L. Liu, C. Li, and Y. Shi, "EILPR: Toward end-to-end irregular license plate recognition based on automatic perspective alignment," *IEEE Trans. Intell. Transp. Syst., vol. 23, no. 3, pp. 2586–2595*, 2022.

[5] P. L. Kumari, R. Tharuni, I. V. S. Sai Vasanth, and M. Vinay Kumar, "Automatic license plate detection using KNN and convolutional neural network," in *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, 2022.

[6] M. Samantaray, A. K. Biswal, D. Singh, D. Samanta, M. Karuppiah, and N. P. Joseph, "Optical character recognition (OCR) based vehicle's license plate recognition system using python and OpenCV," in 2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2021.

[7] A. Firasanti, T. E. Ramadhani, M. A. Bakri, and E. A. Zaki Hamidi, "License plate detection using OCR method with raspberry pi," in *2021 15th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, 2021.

[8] T. Bhagat and R. Thakur, "Automatic recognition of license plates," in *2021 International Conference on Emerging Techniques in Computational Intelligence (ICETCI)*, 2021.

[9] W. Thumthong, P. Meesud, and P. Jarupunphol, "Automatic detection and recognition of Thai vehicle license plate from CCTV images," in *2021 13th International Conference on Information Technology and Electrical Engineering (ICITEE)*, 2021.

[10] P. Mukhija, P. K. Dahiya, and P. Priyanka, "Challenges in automatic license plate recognition system: An Indian scenario," in *2021 Fourth International Conference on Computational Intelligence and Communication Technologies (CCICT)*, 2021.

[11] J.-Y. Sung, S.-B. Yu, and S.-H. P. Korea, "Real-time automatic license plate recognition system using YOLOv4," in *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, 2020.

[12] Joshua, J. Hendryli, and D. E. Herwindiati, "Automatic license plate recognition for parking system using convolutional neural networks," in *2020 International Conference on Information Management and Technology (ICIMTech)*, 2020.

[13] A. Tourani, A. Shahbahrami, S. Soroori, S. Khazaee, and C. Y. Suen, "A robust deep learning approach for automatic Iranian vehicle license plate detection and recognition for surveillance systems," *IEEE Access*, vol. 8, pp. 201317–201330, 2020.

[14] S. H. I. Siqi, L. I. Nanting, M. A. Yanjun, and Z. Liping, "Robust recognition of truck license plate in mine environment," in *2020 International Conference on Computer Engineering and Intelligent Control (ICCEIC)*, 2020.

[15] F. N. M. Ariff, A. S. A. Nasir, H. Jaafar, and A. N. Zulkifli, "Character segmentation for automatic vehicle license plate recognition based on fast K-means clustering," in *2020 IEEE 10th International Conference on System Engineering and Technology (ICSET)*, 2020.

APPENDIX

Fig A.1 Input Image

Fig A.2 License Plate detection by CNN

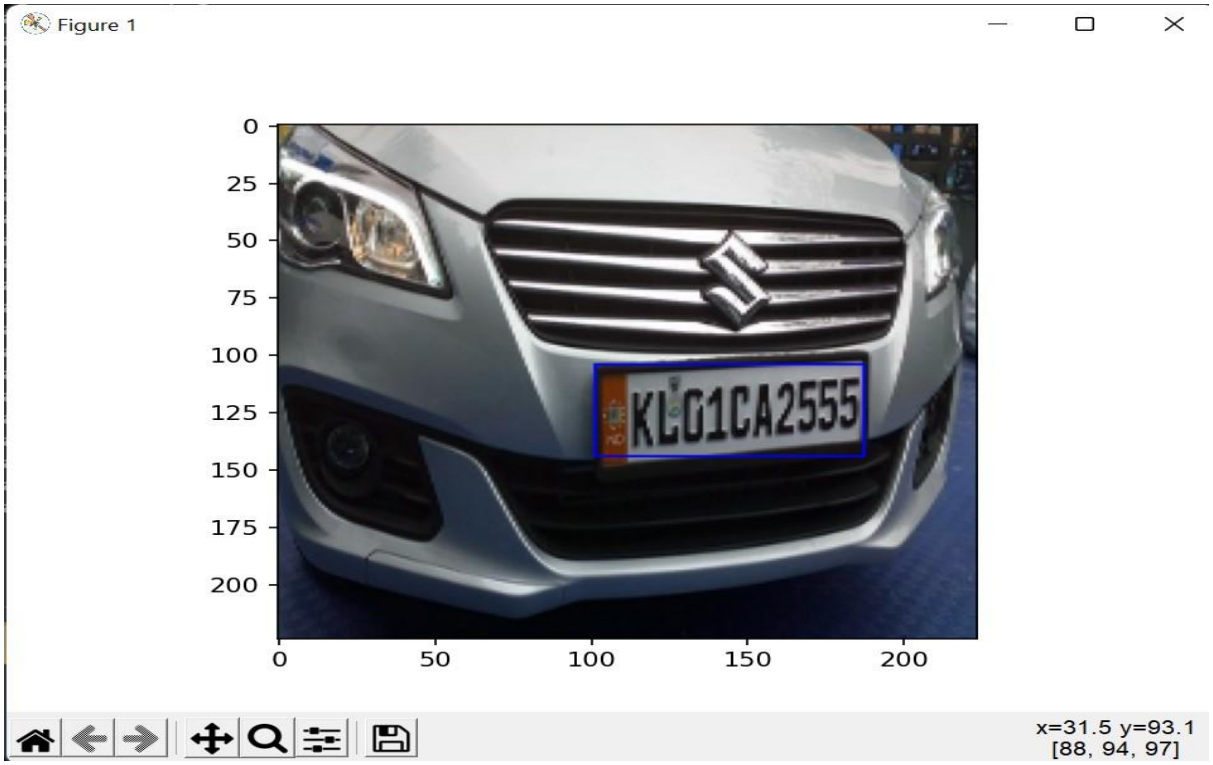

Fig A.3 License Plate detection by VGG16

Fig A.4 License Plate detection by VGG19

Fig A.5 License Plate detection by YOLOV3(a)

Fig A.6 License Plate detection by YOLOV3(b)