

# **AUTOMATED IDENTIFICATION OF VEHICLE USING DEEP LEARNING NEURAL NETWORK**

Submitted in partial fulfillment of the requirements for the award of Bachelor of  
Engineering Degree in Electronics and Communication Engineering

By

**DAMPURU SAI SRUTHIK REDDY (38130050)**

**MADHI TARUN (38130119)**



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION  
ENGINEERING  
SCHOOL OF ELECTRICAL AND ELECTRONICS ENGINEERING**

**SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(DEEMED TO BE UNIVERSITY)**

**Accredited with grade "A" by NAAC**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119**

**MAY 2022**



**SATHYABAMA  
INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)**  
Accredited with "A" grade by NAAC  
**Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119**  
**[www.sathyabama.ac.in](http://www.sathyabama.ac.in)**



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION  
ENGINEERING**

**BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of DAMPURU SAI SRUTHIK REDDY (38130050) and MADHI TARUN (38130119) who carried out the project entitled "AUTOMATED VEHICLE IDENTIFICATION USING DEEP LEARNING NEURAL NETWORK" under my supervision from December 2021 to March 2022.

**Internal Guide**  
**Dr. M. SUGADEV, M Tech, Ph.D.**

**Head of the Department**  
**Dr. T. RAVI M.E., Ph.D.,**

---

**Submitted for Viva voce Examination held on**

**Internal Examiner**

**External Examiner**

## **DECLARATION**

We, DAMPURU SAI SRUTHIK REDDY (38130050) and MADHI TARUN (38130119) hereby declare that the Project Report entitled “AUTOMATED IDENTIFICATION OF VEHICLE DETECTION USING DEEP LEARNING NEURAL NETWORK” done by us under the guidance of Dr. M.SUGADEV, M Tech, Ph D, Associate professor, Dept. of Electronics and Communication Engineering, SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

**DATE:**

**PLACE:**

**SIGNATURE OF CANDIDATE**

## ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to the Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. N. M. NANDHITHA, M.E., Ph.D. Dean, School of Electrical and Electronics Engineering** and **Dr. T. RAVI, M.E., Ph.D. Head of the Department, Department of Electronics and Communication Engineering** for providing us necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to our Project Guide **Dr.M.SUGADEV. M.Tech., Ph.D.**, Associate professor, Dept. of Electronics and Communication Engineering, for his valuable guidance, suggestions and constant encouragement for the successful completion of our project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Electronics and Communication Engineering who were helpful in many ways for the completion of the project.

## **ABSTRACT**

Vehicles have been a big part of many lives; from the time it is invented and as it increases popularity in the 20th century. Though they offer the benefit of convenience, they also have certain negative effects as they add to air pollution and global warming, as well as risks when they are not handled properly. In recent years, the number of vehicles on the road is rapidly increasing and it causes different major concerns. One of the major effects of this increasing volume of vehicles is the traffic congestion it caused on our roads especially in the urban areas. This traffic congestion became one of the major problems in many cities in the world including Metro Manila, Philippines. Many options are discussed and implemented by the traffic management but it seems that it is still unsolved. In recent years, traffic congestions became unpredictable, there are parts of the cities that don't experience traffic congestion then suddenly traffic builds up to that area. Also, traffic congestion might happen every hour of the day.

With this concern, the study proposed a system for vehicle headlight recognition for on-road vehicle detection and counting. This study focused on the detection of the headlight of every vehicle that will be seen on the perimeter of the installed camera. The system can detect headlight vehicles during daytime and nighttime as we trained the AI to recognized the headlights in these scenarios. Possible applications of this system can be in monitoring the volume of vehicles within the area and it can be used by traffic management authority in monitoring the build-up of traffic or traffic situation in an area so that they can provide an immediate solution, such as re-routing, one-way street conversion

# CONTENTS

| CHAPTER NO | TITLE                                                 | PAGE NO |
|------------|-------------------------------------------------------|---------|
|            | <b>ABSTRACT</b>                                       | v       |
|            | <b>LIST OF FIGURES</b>                                | viii    |
|            | <b>LIST OF TABLES</b>                                 | viii    |
| <b>1</b>   | <b>INTRODUCTION</b>                                   |         |
| 1.1        | INTRODUCTION                                          | 1       |
| 1.2        | DEEP LEARNING                                         | 2       |
| 1.3        | CONVOLUTION NEURAL<br>NETWORK (CNN)                   | 4       |
| 1.4        | RESNET34 ARCHITECTURE                                 | 7       |
| <b>2</b>   | <b>LITERATURE SURVEY</b>                              |         |
| 2.1        | LITERATURE SURVEY                                     | 10      |
| <b>3</b>   | <b>AIM AND SCOPE OF THE<br/>PRESENT INVESTIGATION</b> |         |
| 3.1        | AIM                                                   | 20      |
| 3.2        | INFERENCE BLOCK DIAGRAM                               | 21      |
| 3.3        | TRAINING BLOCK DIAGRAM                                | 22      |
| 3.4        | CONVOLUTION 1                                         | 24      |
| 3.5        | RESNET34 LAYERS                                       | 25      |
| 3.6        | BLOCK 1                                               | 31      |
| 3.7        | PATTERNS                                              | 33      |
| 3.8        | BUILDING BLOCKS OF CNN                                | 36      |
| 3.9        | TENSOR FLOW                                           | 36      |
| 3.10       | EPOCHS                                                | 36      |

|          |                              |    |
|----------|------------------------------|----|
| 3.11     | DATA SETS                    | 37 |
| <b>4</b> | <b>RESULT AND DISCUSSION</b> |    |
| 4.1      | TRAINING CLASSES             | 39 |
| 4.2      | RESHAPING AND NORMALIZING    | 41 |
| 4.3      | INITIALIZATION OF RESNET34   | 42 |
| 4.4      | OUTPUT                       | 43 |
| 4.5      | GRAPH                        | 44 |
| 4.6      | MISS PREDICTION OUTPUT       | 45 |
| 4.7      | CONFIDENCE OUTPUT            | 46 |
| <b>5</b> | <b>CONCLUSION</b>            |    |
| 5.1      | CONCLUSION                   | 49 |
|          | REFERENCES                   | 50 |

## LIST OF FIGURES AND TABLES

| FIGURE NO | NAME                        | PAGE NO |
|-----------|-----------------------------|---------|
| 3.1       | INFERENCE BLOCK DIAGRAM     | 21      |
| 3.2       | TRAINING BLOCK DIAGRAM      | 22      |
| 3.3       | RESIDUAL UNIT               | 24      |
| 3.4       | RESNET34 ARCHITECTURE       | 25      |
| 3.5       | CNN LAYERS                  | 26      |
| 3.6       | FLOW OF CNN                 | 27      |
| 3.7       | DEEP LAYERS OF RESNET 34    | 29      |
| 3.8       | CONVOLUTION 1               | 30      |
| 3.9       | MAX POOLING                 | 30      |
| 3.10      | LAYER 1-BLOCK 1-OPERATION 1 | 32      |
| 3.11      | LAYER 1-BLOCK 1             | 32      |
| 3.12      | LAYER 1                     | 33      |
| 3.13      | LAYER 2-BLOCK 1-OPRATION 1  | 34      |
| 3.14      | PROJECTION SHORTCUT         | 34      |
| 3.15      | LAYER 2-BLOCK 1             | 35      |
| 3.16      | LAYER 2                     | 35      |
| 4.1       | CAR DATASET                 | 39      |
| 4.2       | TRAINED DATASET             | 40      |
| 4.3       | RESHAPING AND NORMALIZATION | 41      |
| 4.4.1     | OUTPUT 1                    | 43      |
| 4.4.2     | OUTPUT 2                    | 43      |
| 4.4.3     | OUTPUT 3                    | 44      |
| 4.5       | GRAPH                       | 45      |
| 4.6.1     | MISS PREDICTION OUTPUT 1    | 45      |
| 4.6.2     | MISS PREDICTION OUTPUT 2    | 46      |

|       |                                                             |    |
|-------|-------------------------------------------------------------|----|
| 4.7.1 | CONFEDENCE OUTPUT 1                                         | 47 |
| 4.7.2 | CONFEDENCE OUTPUT 2                                         | 47 |
| 3.1   | Sizes of outputs and convolutional<br>kernels for ResNet 34 | 26 |

# **CHAPTER 1**

## **INTRODUCTION**

Vehicle use has now become an integral part of our lives. These vehicles provide us with convenience parts of the country. From the invention of the vehicle until it became popular in the 20 century, the increase in the number of vehicle on our roads dramatically, especially in urban areas.

In this technology-driven and fast-paced world, constantly managing your professional and personal lives can be hectic. And, if you do not have your own mode of transportation to get from one place to another in a fast and easier manner, then life can become more difficult. In order to make your life much easier, it is best that you find a more reliable and easier method of transportation. In this article, you will be able to know the reasons why cars are very important to our lives and why you should really invest in one in a wise manner.

One of the most important reasons why cars have already become more important and prevalent is because it's a very easy way of transporting from one point to another. Getting from location A to B has actually never been easier and more convenient than ever before. As a matter of fact, if you have a car of your own, you do not need to rely on other public means of transportation anymore for your commutes every day, and you can also be able to enjoy the independence, liberty, and comfort that comes with your own car.

Though they offer the benefit of convenience, they also have certain negative effects as they add tot air pollution and global warming, as well risks when they are not handled properly. In recent years, with the growing number of cars on our roads, one of its main consequences is he occurrence of severe traffic congestion. And traffic congestion nowadays become unpredictable, traffic can build up anytime and anywhere

In recent years, with the growing number of cars on our roads, one of its main consequences is he occurrence of severe traffic congestion. And traffic congestion

nowadays become unpredictable, traffic can build up anytime and anywhere. In recent years, with the growing number of cars on our roads, one of its main consequences is the occurrence of severe traffic congestion. And traffic congestion nowadays become unpredictable, traffic can build up anytime and anywhere.

## **1.1 DEEP LEARNING**

Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised. Deep-learning architectures such as deep neural networks, deep belief networks, deep reinforcement learning, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, climate science, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance

The adjective "deep" in deep learning refers to the use of multiple layers in the network. Early work showed that a linear perceptron cannot be a universal classifier, but that a network with a non-polynomial activation function with one hidden layer of unbounded width can. Deep learning is a modern variation which is concerned with an unbounded number of layers of bounded size, which permits practical application and optimized implementation, while retaining theoretical universality under mild conditions. In deep learning the layers are also permitted to be heterogeneous and to deviate widely from biologically informed connectionist models, for the sake of efficiency, trainability and understandability, whence the "structured" part.

Most modern deep learning models are based on artificial neural networks, specifically convolutional neural networks (CNN)s, although they can also include propositional formulas or latent variables organized layer-wise in deep generative models such as the nodes in deep belief networks and deep Boltzmann machines.

In deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. In an image recognition application, the raw input may be a matrix of pixels; the first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face. Importantly, a deep learning process can learn which features to optimally place in which level *on its own*. This does not eliminate the need for hand-tuning; for example, varying numbers of layers and layer sizes can provide different degrees of abstraction.

The word "deep" in "deep learning" refers to the number of layers through which the data is transformed. More precisely, deep learning systems have a substantial *credit* assignment path (CAP) depth. The CAP is the chain of transformations from input to output. CAPs describe potentially causal connections between input and output. For a feedforward neural network, the depth of the CAPs is that of the network and is the number of hidden layers plus one (as the output layer is also parameterized). For recurrent neural networks, in which a signal may propagate through a layer more than once, the CAP depth is potentially unlimited. No universally agreed-upon threshold of depth divides shallow learning from deep learning, but most researchers agree that deep learning involves CAP depth higher than 2. CAP of depth 2 has been shown to be a universal approximate in the sense that it can emulate any function. Beyond that, more layers do not add to the function approximate ability of the network. Deep models ( $CAP > 2$ ) are able to extract better features than shallow models and hence, extra layers help in learning the features effectively.

Deep learning architectures can be constructed with a greedy layer-by-layer method. Deep learning helps to disentangle these abstractions and pick out which features improve performance. For supervised learning tasks, deep learning methods eliminate feature engineering, by translating the data into compact intermediate representations akin to principal components, and derive layered structures that remove redundancy in representation.

Deep learning algorithms can be applied to unsupervised learning tasks. This is an important benefit because unlabelled data are more abundant than the labelled data. Examples of deep structures that can be trained in an unsupervised manner

are deep belief networks. Deep neural networks are generally interpreted in terms of the universal approximation theorem or probabilistic inference.

The classic universal approximation theorem concerns the capacity of feedforward neural networks with a single hidden layer of finite size to approximate continuous functions. In 1989, the first proof was published by George Cybenko for sigmoid activation functions and was generalised to feed-forward multi-layer architectures in 1991 by Kurt Hornik. Recent work also showed that universal approximation also holds for non-bounded activation functions such as the rectified linear unit.

The universal approximation theorem for deep neural networks concerns the capacity of networks with bounded width but the depth is allowed to grow. Lu et al. proved that if the width of a deep neural network with ReLU activation is strictly larger than the input dimension, then the network can approximate any Lebesgue integrable function; If the width is smaller or equal to the input dimension, then a deep neural network is not a universal approximator.

The probabilistic interpretation derives from the field of machine learning. It features inference as well as the optimization concepts of training and testing, related to fitting and generalization, respectively. More specifically, the probabilistic interpretation considers the activation nonlinearity as a cumulative distribution function. The probabilistic interpretation led to the introduction of dropout as regularize in neural networks. The probabilistic interpretation was introduced by researchers including Hopfield, Widrow and Narendra and popularized in surveys such as the one by Bishop.

## **1.2 CONVOLUTIONAL NEURAL NETWORK**

In deep learning, a convolutional neural network (CNN) is a class of Artificial Neural Network(ANN), most commonly applied to analyse visual imagery. They are also known as Shift Invariant or Space Invariant Artificial Neural Networks (SIANN), based on the shared-weight architecture of the convolution kernels or filters that slide along input features and provide translation equi-variant responses known as feature maps. Counter-intuitively, most convolutional neural networks are only equi-variant, as opposed to invariant, to translation.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "full connectivity" of these networks make them prone to overfitting data. Typical ways of regularization, or preventing overfitting, include: penalizing parameters during training (such as weight decay) or trimming connectivity (skipped connections, dropout, etc.) CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in their filters. Therefore, on a scale of connectivity and complexity, CNNs are on the lower extreme.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns to optimize the filters (or kernels) through automated learning, whereas in traditional algorithms these filters are hand-engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage.

In a CNN, the input is a tensor with a shape: (number of inputs) x (input height) x (input width) x (input channels). After passing through a convolutional layer, the image becomes abstracted to a feature map, also called an activation map, with shape: (number of inputs) x (feature map height) x (feature map width) x (feature map channels). Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus. Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features and classify data, this architecture is generally impractical for larger inputs such as high resolution images. It would require a very high number of neurons, even in a shallow architecture, due to the large input size of images, where each pixel is a relevant input feature. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10,000 weights for *each* neuron in the second layer. Instead,

convolution reduces the number of free parameters, allowing the network to be deeper. For example, regardless of image size, using a  $5 \times 5$  tiling region, each with the same shared weights, requires only 25 learnable parameters. Using regularized weights over fewer parameters avoids the vanishing gradients and exploding gradients problems seen during backpropagation in traditional neural networks. Furthermore, convolutional neural networks are ideal for data with a grid-like topology (such as images) as spatial relations between separate features are taken into account during convolution and/or pooling.

Convolutional networks may include local and/or global pooling layers along with traditional convolutional layers. Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, tiling sizes such as  $2 \times 2$  are commonly used. Global pooling acts on all the neurons of the feature map. There are two common types of pooling in popular use: max and average. *Max pooling* uses the maximum value of each local cluster of neurons in the feature map, while *average pooling* takes the average value. Fully connected layers connect every neuron in one layer to every neuron in another layer. It is the same as a traditional multilayer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images. In neural networks, each neuron receives input from some number of locations in the previous layer. In a convolutional layer, each neuron receives input from only a restricted area of the previous layer called the neuron's *receptive field*. Typically, the area is a square (e.g. 5 by 5 neurons). Whereas, in a fully connected layer, the receptive field is the *entire previous layer*. Thus, in each convolutional layer, each neuron takes input from a larger area in the input than previous layers. This is due to applying the convolution over and over, which takes into account the value of a pixel, as well as its surrounding pixels. When using dilated layers, the number of pixels in the receptive field remains constant, but the field is more sparsely populated as its dimensions grow when combining the effect of several layers.

Each neuron in a neural network computes an output value by applying a specific function to the input values received from the receptive field in the previous layer. The function that is applied to the input values is determined by a vector of

weights and a bias (typically real numbers). Learning consists of iteratively adjusting these biases and weights.

The vector of weights and the bias are called *filters* and represent particular features of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons can share the same filter. This reduces the memory footprint because a single bias and a single vector of weights are used across all receptive fields that share that filter, as opposed to each receptive field having its own bias and vector weighting

### **1.3 RESNET ARCHITECTURE**

Over the last few years, there have been a series of breakthroughs in the field of Computer Vision. Especially with the introduction of deep Convolutional neural networks, we are getting state of the art results on problems such as image classification and image recognition. So, over the years, researchers tend to make deeper neural networks (adding more layers) to solve such complex tasks and to also improve the classification/recognition accuracy. But, it has been seen that as we go adding on more layers to the neural network, it becomes difficult to train them and the accuracy starts saturating and then degrades also. Here Resnet comes into rescue and helps solve this problem. In this article, we shall know more about Resnet and its architecture.

ResNet, short for Residual Network is a specific type of neural network that was introduced in 2015 by Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun in their paper “Deep Residual Learning for Image Recognition”. The ResNet models were extremely successful which you can guess from the following:

Won 1st place in the ILSVRC 2015 classification competition with a top-5 error rate of 3.57% (An ensemble model). Won the 1st place in ILSVRC and COCO 2015 competition in ImageNet Detection, ImageNet localization, Coco detection and Coco segmentation.

Replacing VGG-16 layers in Faster R-CNN with ResNet-101. They observed relative improvements of 28%. Efficiently trained networks with 100 layers and 1000 layers also. Mostly in order to solve a complex problem, we stack some additional layers in the Deep Neural Networks which results in improved accuracy and

performance. The intuition behind adding more layers is that these layers progressively learn more complex features. For example, in case of recognising images, the first layer may learn to detect edges, the second layer may learn to identify textures and similarly the third layer can learn to detect objects and so on. But it has been found that there is a maximum threshold for depth with the traditional Convolutional neural network model. Here is a plot that describes error% on training and testing data for a 20-layer Network and 56 layers Network.

We can see that error% for 56-layer is more than a 20-layer network in both cases of training data as well as testing data. This suggests that with adding more layers on top of a network, its performance degrades. This could be blamed on the optimization function, initialization of the network and more importantly vanishing gradient problem. You might be thinking that it could be a result of overfitting too, but here the error% of the 56-layer network is worst on both training as well as testing data which does not happen when the model is overfitting

This problem of training very deep networks has been alleviated with the introduction of ResNet or residual networks and these Resnets are made up from Residual Blocks. The very first thing we notice to be different is that there is a direct connection which skips some layers (may vary in different models) in between. This connection is called 'skip connection' and is the core of residual blocks. Due to this skip connection, the output of the layer is not the same now. Without using this skip connection, the input 'x' gets multiplied by the weights of the layer followed by adding a bias term.

Next, this term goes through the activation function and we get our output as  $H(x)$ .

$$H(X)=F (wx + b) \quad (1.1)$$

Now with the introduction of skip connection, the output is changed to  $H(X)=f(x)+x$ . There appears to be a slight problem with this approach when the dimensions of the input vary from that of the output which can happen with convolutional and pooling layers. In this case, when dimensions of  $f(x)$  are different

from  $x$ , we can take two approaches. The skip connection is padded with extra zero entries to increase its dimensions. The projection method is used to match the dimension which is done by adding  $1 \times 1$  convolutional layers to input. In such a case, the output is:  $H(x) = f(x)w_1.x$ . Here we add an additional parameter  $w_1$  whereas no additional parameter is added when using the first approach.

## **CHAPTER 2**

### **LITERATURE SURVEY**

Many people done projects on car detection using different ways. Here we have collected some references. And we came up with a project which detects vehicle model using deep learning neural network. Here we listed some of the references related to vehicle detection.

Sania Bhatt (2018) et al. presented a work on Simultaneously Detecting and Counting Dense Vehicles from Drone Images. Traffic Analysis has been a problem that city planners have dealt with for years. Smarter ways are being developed to analyze traffic and streamline the process. Analysis of traffic may account for the number of vehicles in an area per some arbitrary time period and the class of vehicles. People have designed such mechanism for decades now but most of them involve use of sensors to detect the vehicles i.e. a couple of proximity sensors to calculate the direction of the moving vehicle and to keep the vehicle count. Even though over the time these systems have matured and are highly effective, they are not very budget friendly.

The problem is such systems require maintenance and periodic calibration. Therefore, this study has purposed a vision based vehicle counting and classification system. The system involves capturing of frames from the video to perform background subtraction in order detect and count the vehicles using Gaussian Mixture Model (GMM) background subtraction then it classifies the vehicles by comparing the contour areas to the assumed values. The substantial contribution of the work is the comparison of two classification methods. Classification has been implemented using Contour Comparison (CC) as well as Bag of Features (BoF) and Support Vector Machine (SVM) method.

Swetha Rameshkumar (2020) et al. presented work on REAL-TIME VEHICLE COUNTING USING COMPUTER VISION. An image is a visual representation of something. In information technology, the term has several usages. An image is a picture that has been created or copied and stored in electronic form. An image can be described in terms of vector graphics or raster graphics. Digital image processing deals with manipulation of digital images through a digital computer. It is a subfield of

signals and systems but focus particularly on images.

DIP focuses on developing a computer system that is able to perform processing on an image. The input of that system is a digital image and the system process that image using efficient algorithms, and gives an image as an output. It allows much wider range of algorithms to be applied to the input image and can avoid problems such as build-up of noise and signal distortion during processing. An image can be classified into the following three types. In this type of processing typically needs input data provided by the computer vision system and acting as a vision sensor and providing a high-level information. Then the video frames which are captured by the surveillance cameras are given as an input video for vehicle detection and counting.

The Background registration technique is used to construct a reliable background image from the accumulated frame difference information. The moving object region is separated from that background region by comparing the current frame with the background image. In background registration, the history of the frame difference mask is considered in constructing and updating the background buffer. If a pixel is marked as changing in the frame difference mask, the corresponding value in the stationary map is cleared to zero. Otherwise, if the pixel is stationary the corresponding value is incremented by one

Aderonke A Oni (2019) et al. presented work on Video-Based Vehicle Counting System for Urban Roads in Nigeria Using Yolo and DCF-CSR Algorithms. This study developed a vehicle counting system based on YOLO detection algorithm developed by YOLO is a state-of-the-art deep-learning-based object detection system built for real-time applications. Unlike other detectors that use localizers or classifiers for detection purposes, YOLO applies a single neural network to a whole image. It works by dividing the image into regions and predicting object bounding boxes and probabilities for each region. The bounding boxes are then weighted by the calculated probabilities. This technique gives the YOLO algorithm an edge over other object detection algorithms in terms of speed. It has been benchmarked to be 1000 times faster than the R-CNN model and 100 times faster than the Fast R-CNN model.

The tracking module in the vehicle counting system uses Discriminative Correlation Filter with Channel and Spatial Reliability (DCF-CSR) tracking algorithm.

DCF-CSR introduces channel and spatial reliability concepts to discriminative correlation filters (DCF) tracking to produce a better tracking algorithm. The spatial reliability map adjusts the filter support to the area of an object suitable for tracking. This makes it possible to enlarge the search region as well as improves tracking of objects that are not rectangular in shape. Reliability scores reflect the quality of the channels of the learned filters and are used as feature weighting coefficients in the localization process. The vehicle counting system was developed using Visual Studio Code (also called VSCode), the Python interpreter and the OpenCV library. VSCode is a free and open-source cross platform text editor developed by Microsoft. It has many great features including syntax highlighting, intelligent code completion, code snippets and refactoring, Git (a version control system) and debugging

Aishwarya Teli (2021) et al. presented work on Video-Based Detection, Counting and Classification of Vehicles using OpenCV. The detection and counting of vehicles have significant role in an intelligent transportation system, especially for traffic management. Traffic issues has become a biggest problem where city planners facing for years.

The detection of moving vehicles more accurately, several computer vision techniques, vehicle counting is done by using a virtual detection zone. Traffic analysis will calculate for the number of vehicles in an area per some arbitrary period of time and classify the vehicles. But the moving vehicles and their detection, tracking, and counting are very critical for flow of traffic in monitoring, planning, and controlling. By analyzing the recorded video of traffic flow sequence from a camera, a video- based solution technology is applied in combination with virtual detector and blob tracking technologies and YOLO are necessary. With this technology we applied Open CV for real time video applications. Such methods help us to detect, track, count and classify the moving vehicles

The main goal of Vehicle detection and counting in traffic video projects is to develop and implement methodology for automatic detection and counting of moving vehicles on highways. Intelligent visual surveillance (IVS) for road vehicles is a key feature to intelligent transportation systems (ITS).

Bo Zhang (2014) work on “A Traffic Surveillance System for Obtaining Comprehensive Information of the Passing Vehicles Based on Instance

Segmentation". The vehicle information acquired by current video-based traffic surveillance methods is usually one fold. In this report, a traffic surveillance system for obtaining comprehensive vehicle information, including type, number of axles, 3D bounding box, speed, length, current driving lane, and traffic volume, is proposed based on instance segmentation which is realized by Mask R-CNN.

The work mainly contains:

An annotated image dataset which contains car, coach, truck, and wheel is established for training Mask R-CNN, The method for identifying the number of axles based on MaskIoU is proposed, The 3D bounding box is generated based on the high-quality vehicle segmentation, which is more accurate than that generated by background subtraction methods, For calculating the vehicle speed and length, the reference points on the road plane are determined by the lane with dashed line and one vanishing point to calculate the homography matrix, In order to keep the obtained vehicle information reliability, the results identified from multi-frames are obtained by means of the tracking method SORT for processing. The proposed system is verified in different scenes. The average recognition accuracies of the vehicle types and number of axles are more than 97% and 88% in three testing videos. The speed errors of more than 90% vehicles are less than 4%. And the ratios of both missing and repetition in vehicle counting are low in general.

KIM GIL SOO (2017) et al. presented work on Vehicle Detection and Counting in High-Resolution Aerial Images Using Convolutional Regression Neural Network. Vehicle detection and counting in aerial images have become an interesting research focus since the last decade. It is important for a wide range of applications, such as urban planning and traffic management. However, this task is a challenging one due to the small size of the vehicles, their different types and orientations, and similarity in their visual appearance, and some other objects, such as air conditioning units on buildings, trash bins, and road marks. Many methods have been introduced in the literature for solving this problem. These methods are either based on shallow learning or deep learning approaches. However, these methods suffer from relatively low precision and recall rate. This paper introduces an automated vehicle detection and counting system in aerial images. The proposed system utilizes convolution neural network to regress a vehicle spatial density map across the aerial image. It has been evaluated on two publicly available data sets, namely, Munich and Overhead Imagery

Research Data Set.

The experimental results show that our proposed system is efficient and effective, and produces higher precision and recall rate than the comparative methods. Vehicle detection and counting is important for many applications such as surveillance, traffic management, and rescue tasks. The ability of on-line monitoring of vehicles distribution in the urban environments prevents traffic jams and congestions which in turn reduces air and noise pollution. In terms of surveillance, the accurate estimation of vehicles in parking lots or roads is essential for making right decisions.

Dr. Ravindra Kumar (2019) et al. presented work on Vehicle Detection and Counting Method Based on Digital Image Processing in Python. Vehicle counting process provides appropriate information about traffic flow, vehicle crash occurrences and traffic peak times in roadways. An acceptable technique to achieve these goals is using digital image processing methods on roadway camera video outputs. This paper presents a vehicle counter classifier based on a combination of different video image processing methods including object detection, edge detection, frame differentiation and the Kalman filter. An implementation of proposed technique has been performed using python programming language. This paper describes the methodology used for image processing for traffic flow counting and classification using different library and algorithm with real time image

Vehicles detection and counting are done a non-intrusive sensor/manual method/video infrared, magnetic, radar, ultrasonic, acoustic, and video imaging sensors and intrusive sensor sensors included pneumatic road tube, piezo-electric sensor, magnetic sensor, and inductive loop. Nonintrusive technology has advantages over intrusive technology which requires closing of traffic lanes and put construction workers in harm's way, stop traffic or a lane closure and non-intrusive sensors are above the roadway surface and don't typically require a stop in traffic or lane closure.

Both types of sensors have advantages and disadvantages. But the accuracy from video or digital counting manually is very high as compared to other technology. However, the image processing is time consuming and requires some automation to save the time for image count and classification. In current era of python type programming language has much addition of image processing and time saving for vehicle detection, counting and classification.

Ryan R. Tejada (2018) et al. presented work on an Augmented Deep Learning Inference Approach of Vehicle Headlight Recognition for On-Road Vehicle Detection and Counting. To create the system, the study used a pre-trained algorithm and datasets. These new datasets underwent custom training to get weights using YOLOv3. YOLOv3 is the latest variant of YOLO (You Only Look Once), a fast and accurate object detection model compared to Faster R-CNN. Then dataset is annotated using Labeling Software, a graphical image annotation tool. by creating a boundary box for the object to be detected and this will define the x and y coordinates of the object. Also, an annotation text file will be created in .xml format (Pascal VOC Format), which consists of the information of the image annotated and this file can be provided by the Labeling.

Once the annotation of the datasets or the image these are trained using YOLOv3. The study used Google Collaborator (Google Collaborator) for the training, evaluation, and testing of the dataset and model and to execute this libraries and dependencies must be downloaded and installed The graphs presented are the generated result during the custom training and experiment. During the training and results, the datasets that underwent custom training are being weighed. The images are being trained and validated many times to generate a good model for the system. On the phase of evaluation and testing of the module, the model generated during training will be evaluated based on the generated MAP (mean average precision). A high MAP can be an indicator of high prediction accuracy. And lastly, for the testing, the selected model with high MAP will be used in sample photos and videos to see if the chosen model accurately detects the object of interest.

King Ngi Ngan (2016) et al. presented work on Simultaneously Detecting and Counting Dense Vehicles from Drone Images. Unmanned aerial vehicles” (UAVs) are now being used for a wide range of surveillance applications. Specifically, the detection of on-ground vehicles from UAV images has attracted significant attention due to its potential in applications such as traffic management, parking lot management, and facilitating rescue operations in disaster zones and rugged terrains.

This paper presents a survey of deep learning techniques for performing on-ground vehicle detection from aerial imagery captured using UAVs (also known as drones). We review the works in terms of their approach to improve accuracy and reduce computation overhead and their optimization objective. We show the similarities and differences of various techniques and also highlight the future

challenges in this area. This survey will benefit researchers in the area of artificial intelligence, traffic surveillance, and applications of UAVs.

Recent years have seen an enormous increase in the use of “unmanned aerial vehicles” (UAVs) for remote sensing and surveillance applications. UAVs afford several advantages such as low cost, on-demand image-collection, and the ability to monitor rugged or harsh terrains and disaster zones. As the transportation system becomes increasingly complex, vehicle detection from UAV images has become extremely important. It can help traffic management, parking lot management, and locating the vehicles stuck in rugged terrains or disaster zones. Vehicle detection from UAV images can also complement on-road vehicle detection and, thus, prove useful for driver assistance systems. Further vehicle detection is usually the first step in many traffic surveillance tasks.

Qi Wang, Zhougyuan (2015) et al. presented work on Fine-grained vehicle recognition in traffic surveillance. The increasingly dense traffic is becoming a challenge in our local settings, urging the need for a better traffic monitoring and management system. Fine-grained vehicle classification appears to be a challenging task as compared to vehicle coarse classification. Exploring a robust approach for vehicle detection and classification into fine-grained categories is therefore essentially required. Existing Vehicle Make and Model Recognition (VMMR) systems have been developed on synchronized and controlled traffic conditions. Need for robust VMMR in complex, urban, heterogeneous, and unsynchronized traffic conditions still remain an open research area. In this paper, vehicle detection and fine grained classification are addressed using deep learning. To perform fine-grained classification with related complexities, local dataset THS-10 having high intra class and low interclass variation is exclusively prepared. The dataset consists of 4250 vehicle images of 10 vehicle models, i.e., Honda City, Honda Civic, Suzuki Alto, Suzuki Bolan, Suzuki Cultus, Suzuki Mehran, Suzuki Ravi, Suzuki Swift, Suzuki Wagon R and Toyota Corolla.

This dataset is available online. Due to having almost no design variation in some make and models over the years, vehicle models are not separated by their year of generation. Two approaches have been explored and analyzed for classification of vehicles i.e., fine-tuning, and feature extraction from deep neural networks. A comparative study is performed, and it is demonstrated that simpler approaches can

produce good results in local environment to deal with complex issues such as dense occlusion and lane departures.

D C. Chuang (2019) et al. presented work on "Overtaking Vehicle Detection Based on Deep Learning and Headlight Recognition. When the daytime is full of light, the region-based convolutional neural networks (RCNN) is used to identify and track vehicles. However, due to the lack of light during the nighttime, the headlights are used for vehicle identification and tracking. When the detected position and driving direction of the vehicle satisfy the overtaking condition, the driver can be immediately warned that there will be an overtaking vehicle. In the experiment, the accuracy of the overtaking vehicle detection (precision) can reach more than 90%. Image-based overtaking vehicle detection or blind spot detection systems mainly use image recognition and computer vision related technology. Traditional methods usually use the shape and texture of vehicles to identify vehicles. In recent years, the method of deep learning can effectively improve the detection accuracy. Currently, there are many methods for deep learning identification such as convolutional neural networks (CNN), deep neural networks (DNN), and region-based convolutional neural networks (RCNN).

In terms of neural networks, multiple images of different vehicles with different angles are used as training samples to increase the recognition rate of RCNN. There is also a refining block used in the fast-RCNN, and the refinement block and the faster-RCNN are combined into a single network to improve the recognition rate. As for vehicle re-identification, a two-branch multi-DNN fusion Siamese neural network (MFSNN) was proposed to fuse the classification outputs of vehicle colors, model and pasted marks [1]. In the aerial images, faster-RCNN can also be used to detect and identify objects to reduce the difficulty of rescue.

H. Leung (2014) et al. presented work on "A Deep Learning-Based Vehicle Detection Approach for Insufficient and Night Time Illumination Conditions. Most object detection models cannot achieve satisfactory performance under night time and other insufficient illumination conditions, which may be due to the collection of data sets and typical labelling conventions. Public data sets collected for object detection are usually photographed with sufficient ambient lighting. However, their labelling conventions typically focus on clear objects and ignore blurry and occluded objects. Consequently, the detection performance levels of traditional vehicle detection techniques are limited in night time environments without sufficient illumination. When objects occupy a small number of pixels and the existence of crucial features is infrequent, traditional convolutional neural networks (CNNs) may suffer from serious information loss due to

the fixed number of convolutional operations. This study presents solutions for data collection and the labelling convention of night time data to handle various types of situations, including in-vehicle detection.

Moreover, the study proposes a specifically optimized system based on the Faster region-based CNN model. The system has a processing speed of 16 frames per second for  $500 \times 375$ -pixel images, and it achieved a mean average precision (MAP) of 0.8497 in our validation segment involving urban night time and extremely inadequate lighting conditions. The experimental results demonstrated that our proposed methods can achieve high detection performance in various night time environments, such as urban night time conditions with insufficient illumination, and extremely dark conditions with nearly no lighting. The proposed system outperforms original methods that have an MAP value of approximately

D L Jiang (2014) et al. presented work on Vehicle detection based on morphology from highway aerial images

Vehicle detection from high resolution aerial images has been studied for many years. However, a robust and efficient vehicle detection method is still challenging. In this paper, a novel and robust method for automatic vehicle detection from highway aerial image was presented. In this method, a GIS road vector map is used to constrain a vehicle detection system to the highway networks. After the structure element is identified, morphological preprocessing method is used to identify candidate vehicles. Experiment is conducted with 0.15 m resolution aerial image. And the result demonstrated that the novel method has an excellent detection performance, thus the method is very promising.

With the development of traffic, there is high demand in traffic monitoring of urban areas. Currently the traffic monitoring is implemented by a lot of ground sensors, like induction loops, bridge sensors and stationary cameras. However, these sensors can only partly acquire the traffic flow on main highway subsets. The traffic on most highway subsets is seldom collected. Hence, area-wide images of the entire highway networks are required to complement these selectively acquired data [1] [2]. Vehicles can be observed clearly on these high resolution aerial images with a ground resolution of 0.15 m. Thus, applications of high-resolution aerial imager to detect vehicle for traffic monitoring are attractive.

U Stilla (2013) et al. presented work on Vehicle detection in very high resolution satellite images of city areas. Current traffic research is mostly based on data from fixed-installed sensors like induction loops, bridge sensors, and cameras. Thereby, the traffic flow on main roads can partially be acquired, while data from the major part of the entire road network are not available. Today's optical sensor systems on satellites provide large-area images with 1-m resolution and better, which can deliver complement information to traditional acquired data. In this paper, we present an approach for automatic vehicle detection from optical satellite images. Therefore, hypotheses for single vehicles are generated using adaptive boosting in combination with features. Additionally, vehicle queues are detected using a line extraction technique since grouped vehicles are merged to either dark or bright ribbons. Utilizing robust parameter estimation, single vehicles are determined within those vehicle queues. The combination of implicit modelling and the use of a priori knowledge of typical vehicle constellation leads to an enhanced overall completeness compared to approaches which are only based on statistical classification techniques. Thus, a detection rate of over 80% is possible with very high reliability.

Furthermore, an approach for movement estimation of the detected vehicle is described, which allows the distinction of moving and stationary traffic. Thus, even an estimate for vehicles' speed is possible, which gives additional information about the traffic condition at image acquisition time.

## CHAPTER 3

### PROPOSED METHODOLOGY

#### 3.1 AIM

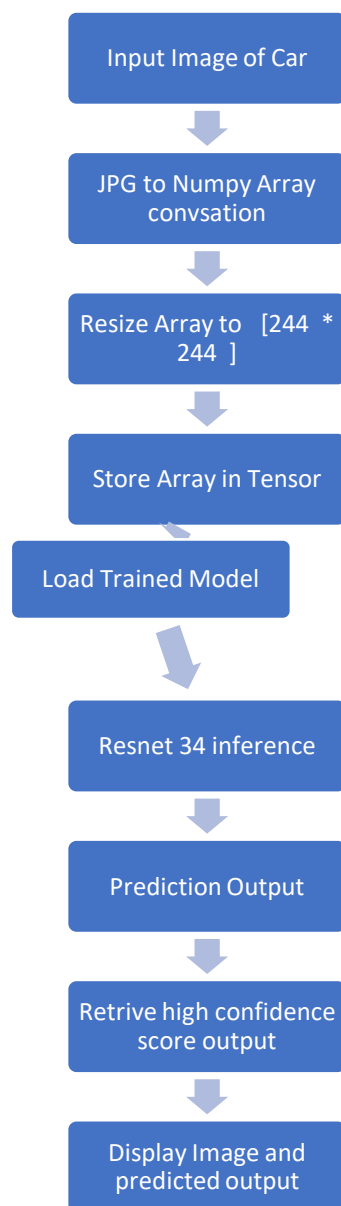
To write a software pipeline to identify vehicles model using data set. In my implementation, I used a Deep Learning approach to image recognition. Specifically, I leveraged the extraordinary power of Convolutional Neural Network (CNNs) to recognize images. Having trained this type of a model, the input's width and height dimensions can be expanded arbitrarily, transforming the output layer's dimensions from 1x1 to a map with an aspect ratio approximately matching that of a new large input.

Here this model classifies and detects whether a vehicle is a car, bus or a bike. This paper creates a convolution neural network from scratch to classify and detect these vehicles using a modern convolution neural network based on fast regions. Mostly in order to solve a complex problem, we stack some additional layers in the Deep Neural Networks which results in improved accuracy and performance. The intuition behind adding more layers is that these layers progressively learn more complex features.

The skip connections in ResNet-34 solve the problem of vanishing gradient in deep neural networks by allowing this alternate shortcut path for the gradient to flow through. The other way that these connections help is by allowing the model to learn the identity functions which ensures that the higher layer will perform at least as good as the lower layer, and not worse. Let me explain this further. So we here use ResNet-34 architecture to get more accuracy. Here we are using 10 epochs for testing. To get more accuracy we can use 20, then can get 5-6 %percent more accuracy.

### 3.2 INFERENCE BLOCK DIAGRAM

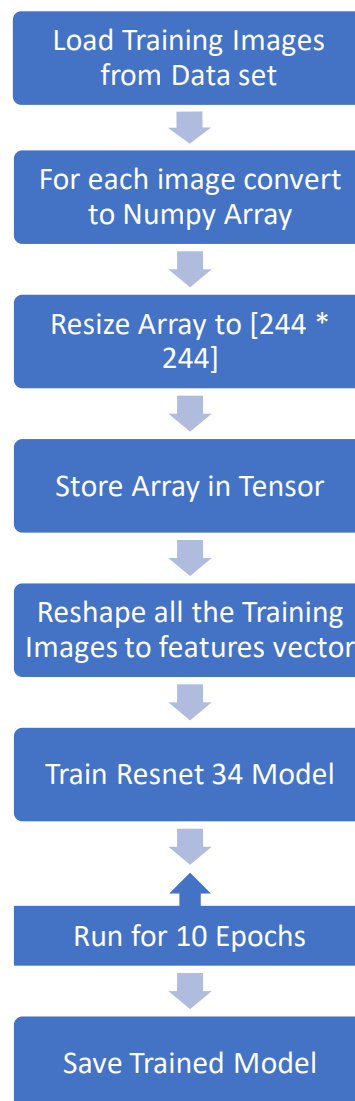
This is inference block diagram of our model, here at first we take images as input and then we convert them into “.jpg” format. After we are resizing the images into [244,244] array and storing in tensor(training). Now we are loading the trained model with Resnet34 inference. There by testing the image to detect its model by giving input as “.jpg” format. Finally, the predicted output will get displayed in the output screen.



**Fig:3.1: Inference Block Diagram**

### 3.3 TRAINING BLOCK DIAGRAM

This is inference block diagram of our model, here at first we take images as input and then we convert them into “.jpg” format. After we are resizing the images into [244,244] array and storing in tensor. Here reshaping all the images of training, testing, validation to features vector. Now by running each epoch to ten resnet34 model will get trained. Now the model is ready for testing.



**Fig:3.2: Training Block Diagram**

### 3.3 CONVOLUTIONAL NEURAL NETWORK

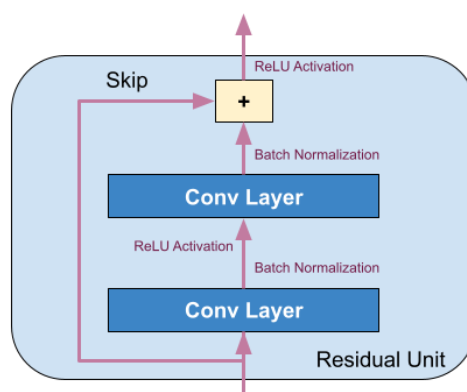
Researchers observed that it makes sense to affirm that “*the deeper the better*” when it comes to convolutional neural networks. This makes sense, since the models should be more capable (their flexibility to adapt to any space increase because they have a bigger parameter space to explore). However, it has been noticed that after some depth, the performance degrades. This was one of the bottlenecks of VGG. They couldn’t go as deep as wanted, because they started to lose generalization capability. Since neural networks are good function approximator, they should be able to easily solve the identity function, where the output of a function becomes the input itself. Following the same logic, if we bypass the input to the first layer of the model to be the output of the last layer of the model, the network should be able to predict whatever function it was learning before with the input added to it.

$$F(x) + x = h(x)$$

The intuition is that learning  $f(x) = 0$  has to be easy for the network

One of the problems Resnets solve is the famous known vanishing gradient. This is because when the network is too deep, the gradients from where the loss function is calculated easily shrink to zero after several applications of the chain rule. This result on the weights never updating its values and therefore, no learning is being performed.

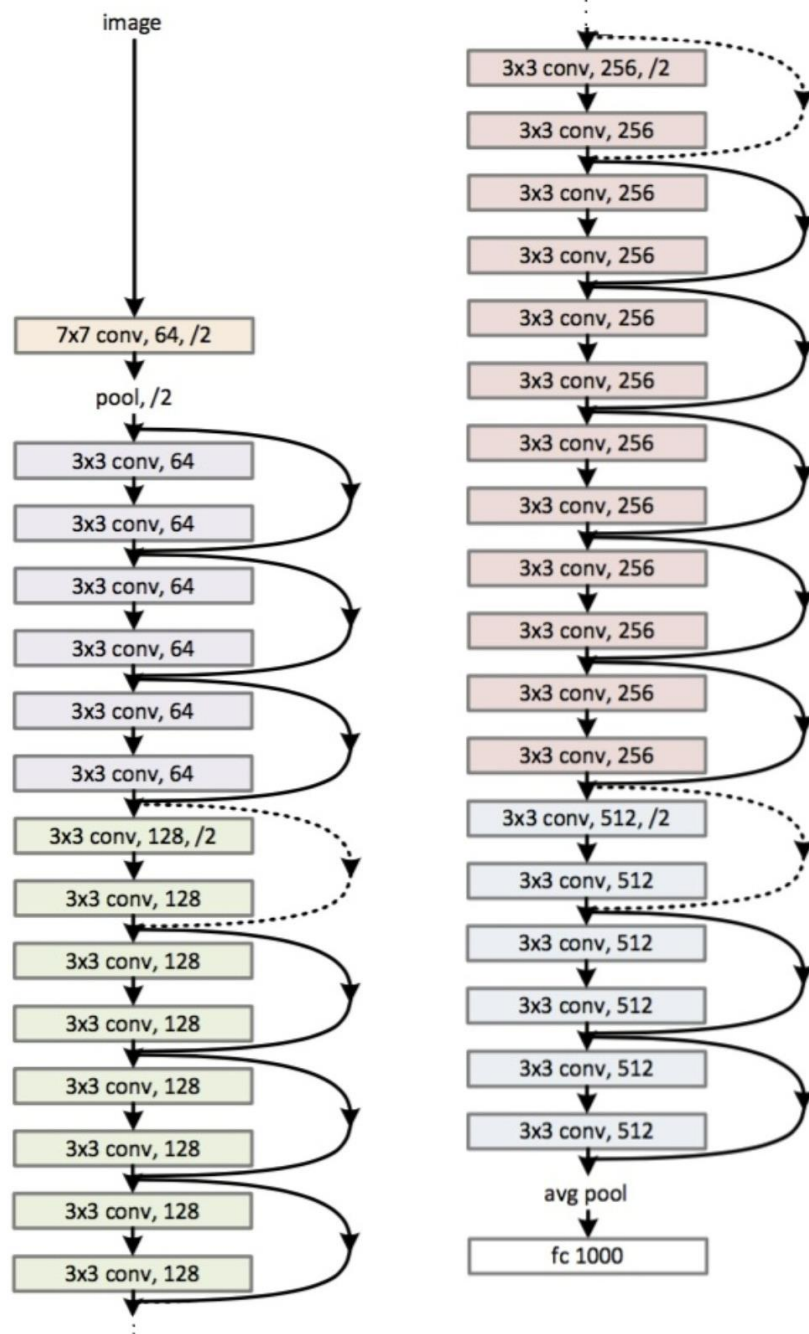
With Resnet, the gradients can flow directly through the skip connections backwards from later layers to initial filters.



**Fig:3.3: General Residual Unit**

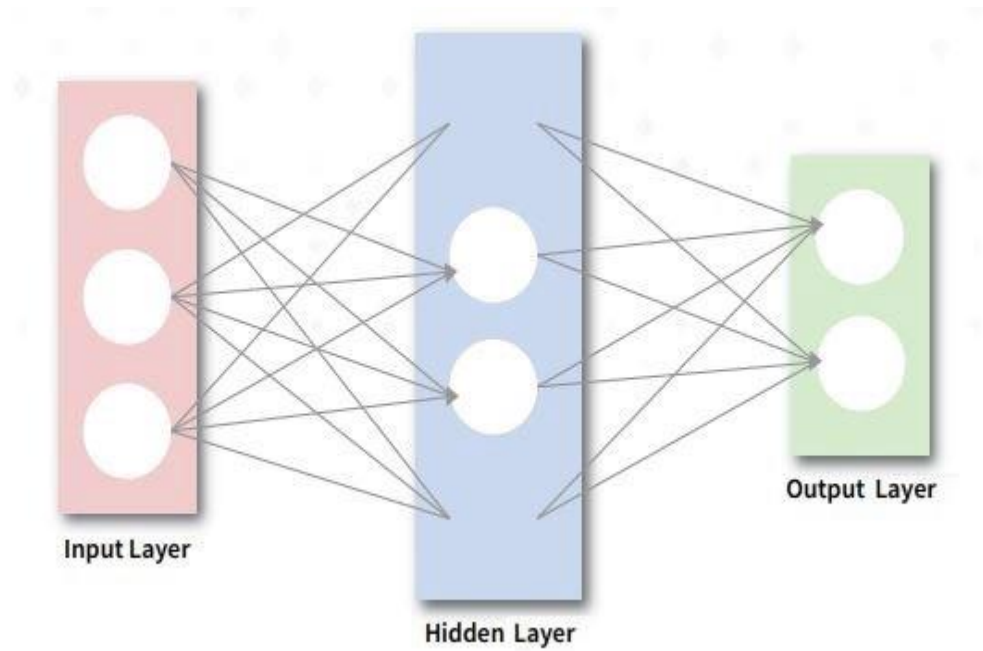
### 3.4 RESNET-34 ARCHITECTURE

34-layer residual



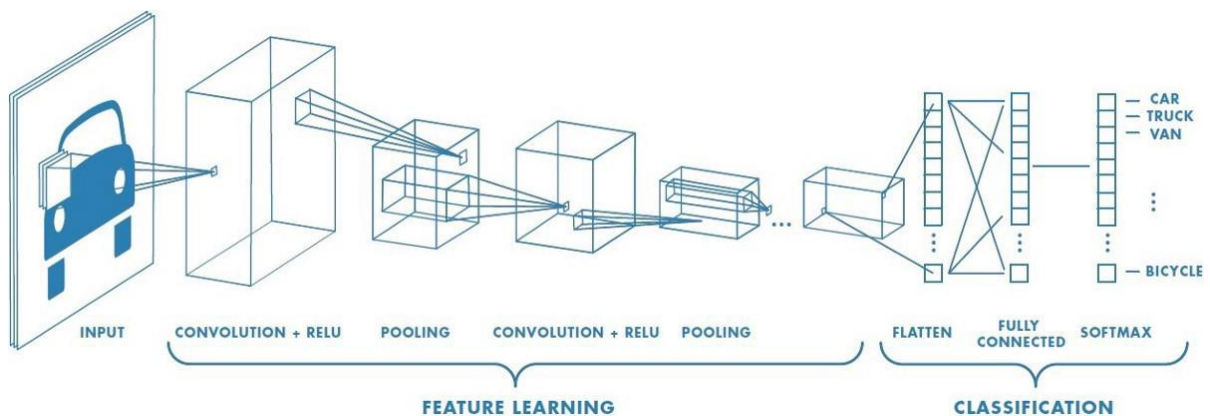
**Fig:3.4: Resnet-34 Architecture**

The pre-trained model here is ResNet (Residual Neutral Network).



**Fig:3.5: CNN**

Artificial Neural Network predicts the outcome by learn through the examples and previous experiences. Artificial Neural Network computations carried out in parallel and create their own representation of information which they receive during the learning time. In the artificial Neural Network, just add in one more layer which is convolutional layer.



**Fig:3.6: Flow of CNN**

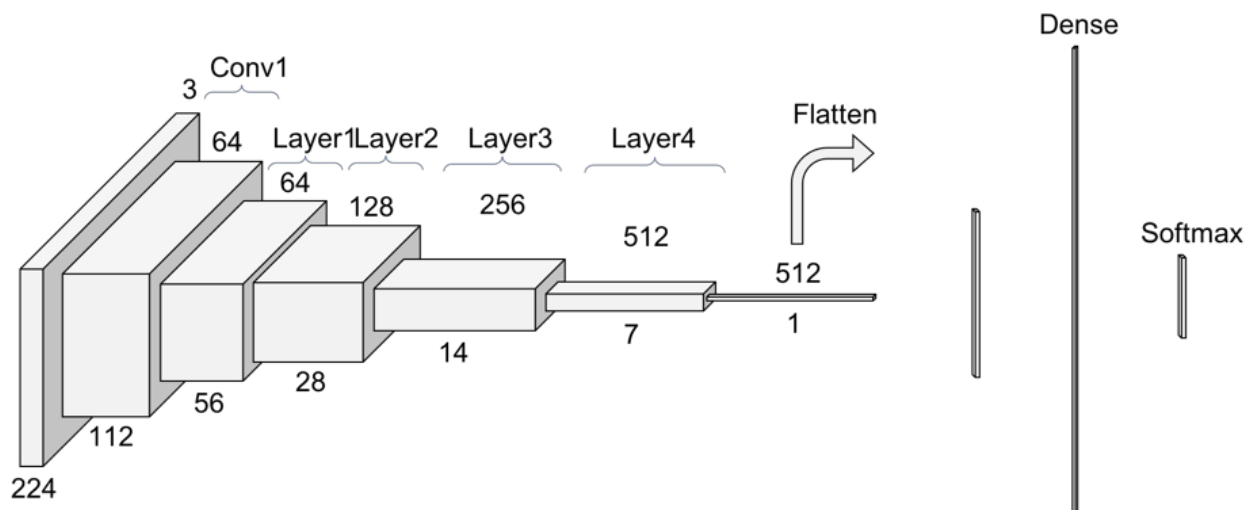
Since ResNet can have variable sizes, depending on how big each of the layers of the model are, and how many layers it has, we will follow the described by the authors in the paper [1] — ResNet 34 — in order to explain the structure after these networks. If you have taken a look at the paper, you will have probably seen some figures and tables like the following ones, that you are struggling to follow. Let's depict those figures by going into the detail of every step. In here we can see that the ResNet (the one on the right) consists on one convolution and pooling step (on orange) followed by 4 layers of similar behavior.

Each of the layers follow the same pattern. They perform 3x3 convolution with a fixed feature map dimension (F) [64, 128, 256, 512] respectively, bypassing the input every 2 convolutions. Furthermore, the width (W) and height (H) dimensions remain constant during the entire layer. The dotted line is there, precisely because there has been a change in the dimension of the input volume (of course a reduction because of the convolution). Note that this reduction between layers is achieved by an increase on the stride, from 1 to 2, at the first convolution of each layer; instead of by a pooling operation, which we are used to see as down samplers. In the table, there is a summary of the output size at every layer and the dimension of the convolutional kernels at every point in the structure.

**TABLE 3.1: Sizes of outputs and convolutional kernels for ResNet 34**

| layer name | output size | 18-layer                                                                    | 34-layer                                                                    | 50-layer                                                                                        | 101-layer                                                                                        | 152-layer                                                                                        |
|------------|-------------|-----------------------------------------------------------------------------|-----------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| conv1      | 112×112     | 7×7, 64, stride 2                                                           |                                                                             |                                                                                                 |                                                                                                  |                                                                                                  |
| conv2_x    | 56×56       | 3×3 max pool, stride 2                                                      |                                                                             |                                                                                                 |                                                                                                  |                                                                                                  |
|            |             | $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$   | $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$   | $\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$    | $\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$     | $\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$     |
| conv3_x    | 28×28       | $\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$ | $\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$ | $\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$  | $\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$   | $\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$   |
| conv4_x    | 14×14       | $\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$ | $\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$ | $\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$ | $\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$ | $\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$ |
| conv5_x    | 7×7         | $\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$ | $\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$ | $\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$ | $\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$  | $\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$  |
|            | 1×1         | average pool, 1000-d fc, softmax                                            |                                                                             |                                                                                                 |                                                                                                  |                                                                                                  |
| FLOPs      |             | 1.8×10 <sup>9</sup>                                                         | 3.6×10 <sup>9</sup>                                                         | 3.8×10 <sup>9</sup>                                                                             | 7.6×10 <sup>9</sup>                                                                              | 11.3×10 <sup>9</sup>                                                                             |

The Figure 6 is the way I prefer to see convolutional models, and from which I will explain every layer. Here we observe how actually the volumes that are going through the model are changing their sizes. This way is easier to understand the mechanism of a particular model, to be able to adjust it to our particular needs — we will see how just changing the dataset forces to change the architecture of the entire model. Also, I will try to follow the notation close to the PyTorch official implementation to make it easier to later implement it on PyTorch. For instance, ResNet on the paper is mainly explained for ImageNet dataset. But the first time I wanted to make an experiment with ensembles of ResNets, I had to do it on CIFAR10. Obviously, since CIFAR10 input images are (32x32) instead of (224x224), the structure of the ResNets need to be modify. If you want to have a control on the modifications to apply to your ResNet, you need to understand the details.



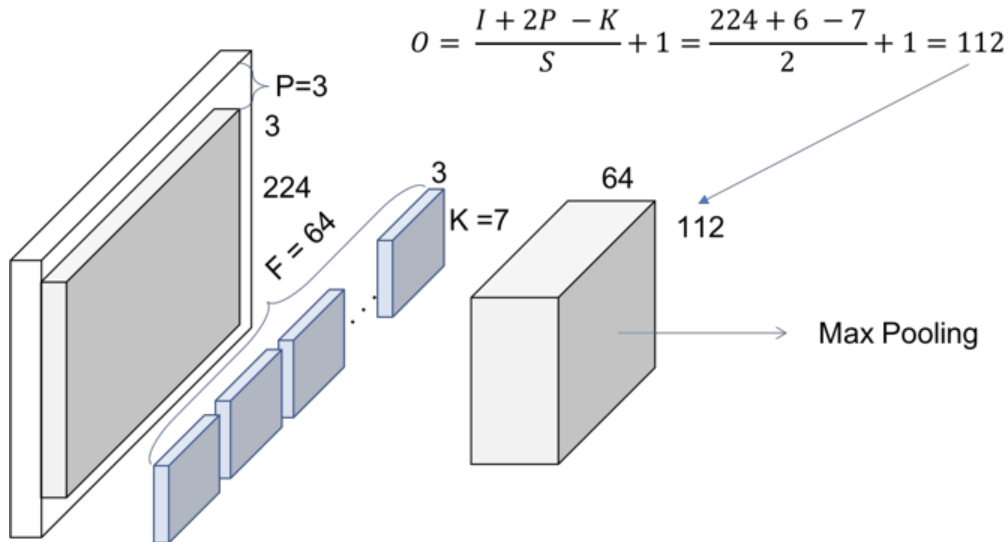
**Fig:3.7: Another look at ResNet 34**

### 3.5 Convolution 1

The first step on the ResNet before entering the common layer behavior is a block called here Conv1 consisting on a convolution + batch normalization + max pooling operation. If you don't remember how convolutions and pooling operations where performed, take a quick look at this draws I made to explain them, since I reused part of them.

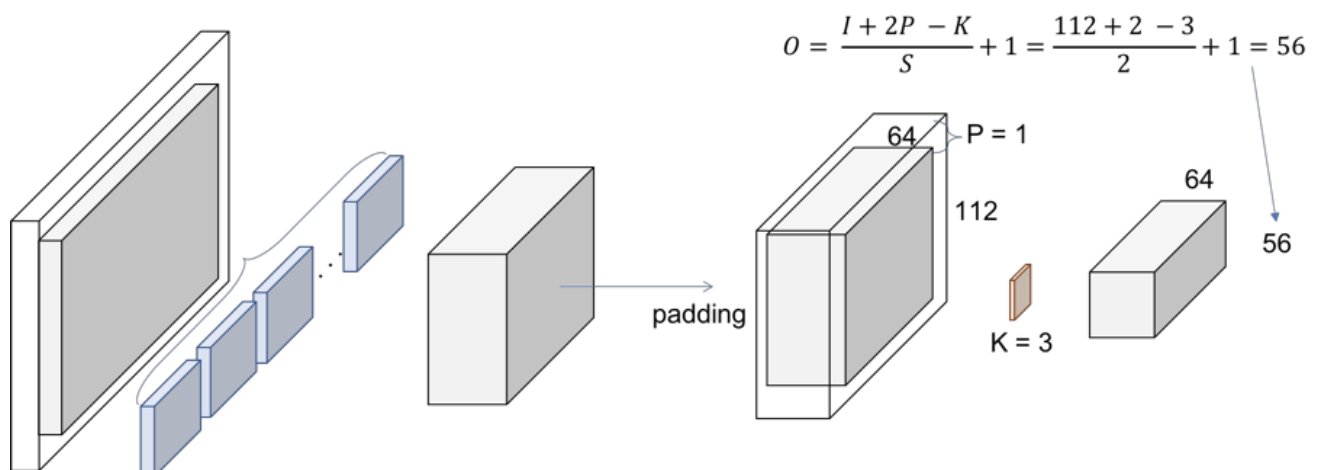
So, first there is a convolution operation. In the Figure 1 we can see that they use a kernel size of 7, and a feature map size of 64. You need to infer that they have

padded with zeros 3 times on each dimension and check it on the PyTorch documentation. Taken this into account, it can be seen in Figure 4 that the output size of that operation will be a (112x122) volume. Since each convolution filter (of the 64) is providing one channel in the output volume, we end up with a (112x112x64) output volume — note this is free of the batch dimension to simplify the explanation.



**Fig:3.8: Conv1 — Convolution**

The next step is the batch normalization, which is an element-wise operation and therefore, it does not change the size of our volume. Finally, we have the (3x3) Max Pooling operation with a stride of 2. We can also infer that they first pad the input volume, so the final volume has the desired dimensions.



**Fig:3.9: Conv1 — Max Pooling**

## 3.5 RESNET LAYERS

So, let's explain this repeating name, block. Every layer of a ResNet is composed of several blocks. This is because when ResNets go deeper, they normally do it by increasing the number of operations within a block, but the number of total layers remains the same — 4. Therefore, in the PyTorch implementation they distinguish between the blocks that includes 2 operations

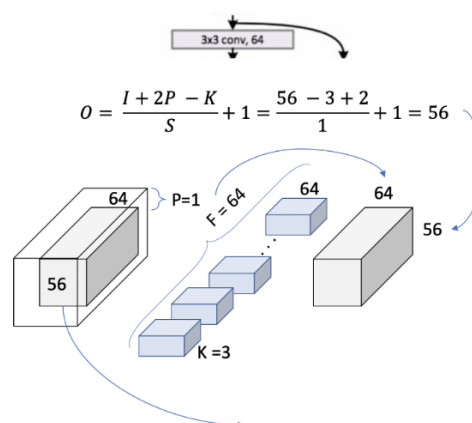
**Basic Block** — and the blocks that include 3 operations

**Bottleneck Block.** Note that normally each of these operations is called layer, but we are using layer already for a group of blocks. We are facing a Basic one now. The input volume is the last output volume from Conv1. Let's see Figure 9 to figure out what is happening inside this block.

## 3.6 Block 1

### 1 convolution

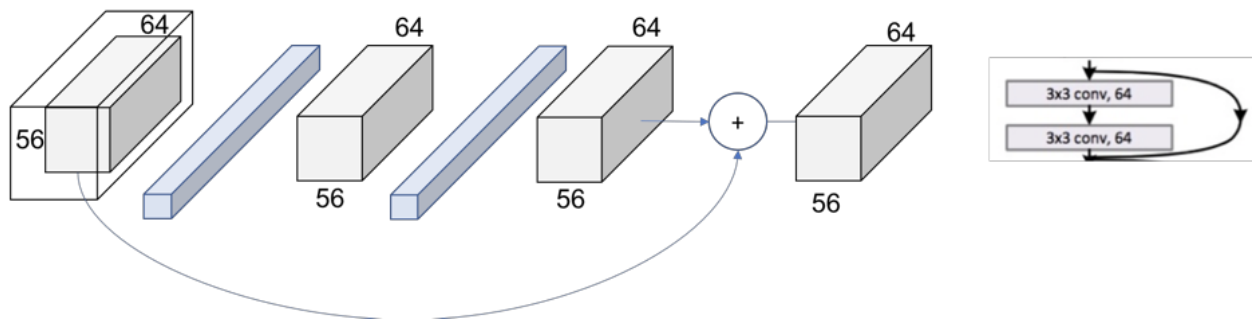
We are replicating the simplified operation for every layer on the paper.



**Fig:3.10: Layer 1, block 1, operation 1**

We can double check now in the table from the paper we are using [3x3, 64] kernel and the output size is [56x56]. We can see how, as we mentioned previously,

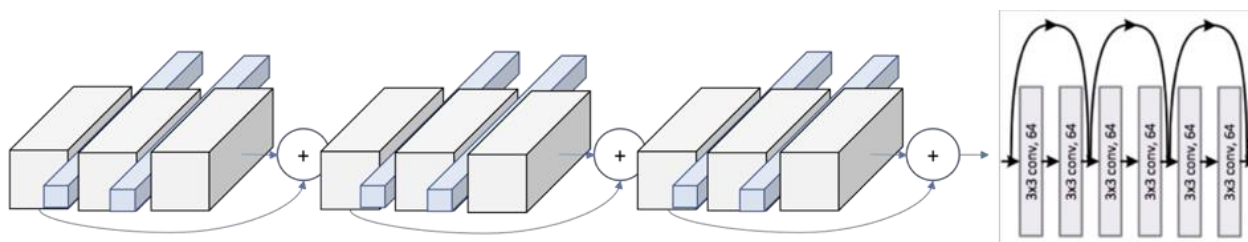
the size of the volume does not change within a block. This is because a padding = 1 is used and a stride of also 1. Let's see how this extends to an entire block, to cover the 2 [3x3, 64] that appears in the table.



**Fig.3.11: Layer 1, block 1**

The same procedure can be expanded to the entire layer then as in Figure 11. Now, we can completely read the whole cell of the table (just recap we are in the 34 layers ResNet at Conv2\_x layer).

We can see how we have the [3x3, 64] x 3 times within the layer.



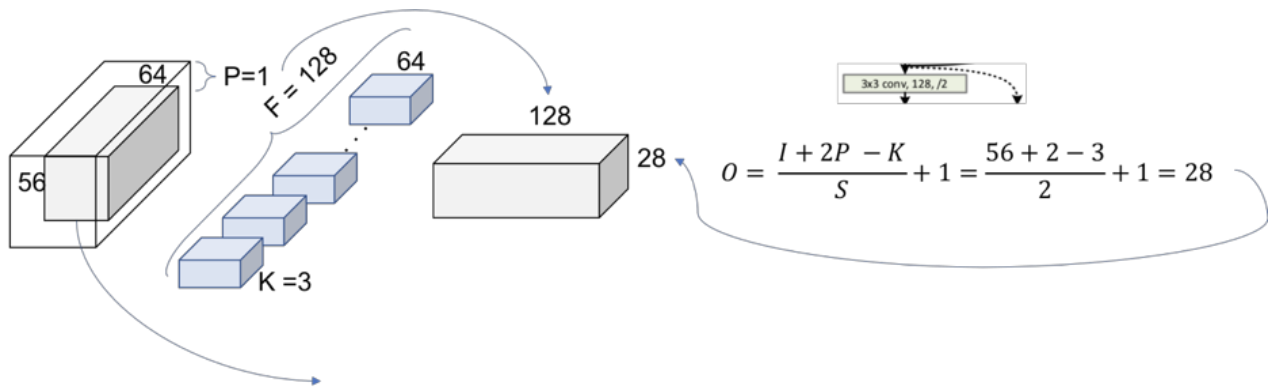
**Fig.3.12: Layer 1**

### 3.7 PATTERNS

The next step is to escalate from the entire block to the entire layer. In the Figure 3 we can see how the layers are differentiable by colors. However, if we look at the first operation of each layer, we see that the stride used at that first one is 5, instead of 4 like for the rest of them. This means that the down sampling of the volume though the network is achieved by increasing the stride instead of a pooling operation like normally CNNs do. In fact, only one max pooling operation is performed in our Conv1 layer, and one average pooling layer at the end of the ResNet, right before the fully connected dense layer in Figure 3.

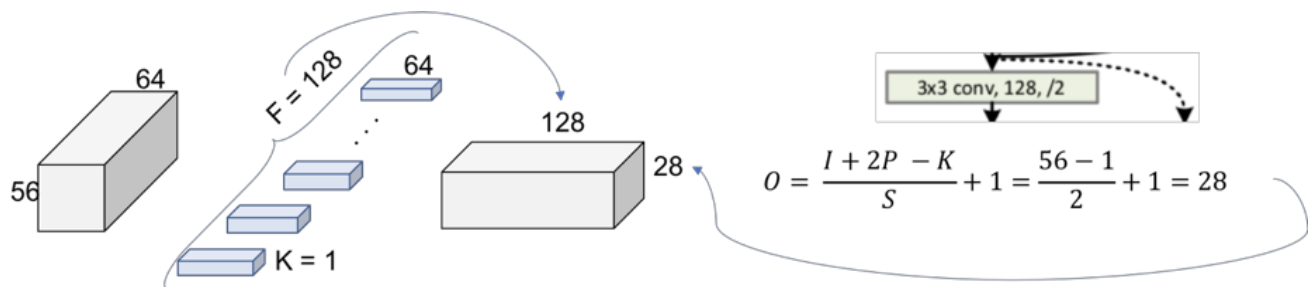
We can also see another repeating pattern over the layers of the ResNet, the dot layer representing the change of the dimensionality. This agrees with what we just said. The first operation of each layer is reducing the dimension, so we also need to resize the volume that goes through the skip connection, so we could add them like we did in Figure 10.

This difference on the skip connections are the so called in the paper as Identity Shortcut *and* Projection Shortcut. The identity shortcut is the one we have already discussed, simply bypassing the input volume to the addition operator. The projection shortcut performs a convolution operation to ensure the volumes at this addition operation are the same size. From the paper we can see that there are 2 options for matching the output size. Either *padding the input volume* or perform *1x1 convolutions*. Here, this second option is shown.



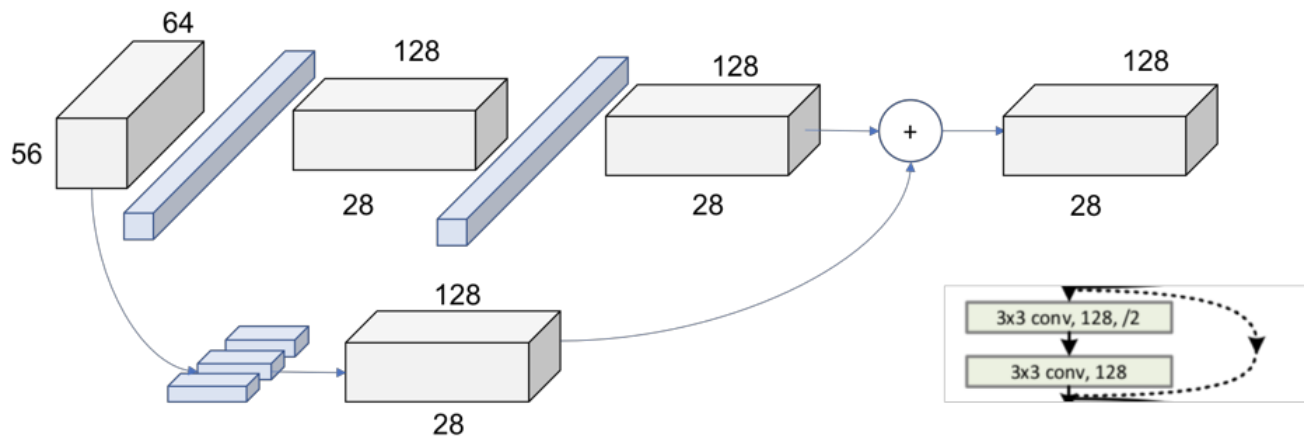
**Fig:3.13: Layer2, Block 1, operation 1**

Figure 9 represents this down sampling performed by increasing the stride to 2. The number of filters is duplicated in an attempt to preserve the time complexity for every operation ( $56 \times 64 = 28 \times 128$ ). Also, note that now the addition operation cannot be performed since the volume got modified. In the shortcut we need to apply one of our down sampling strategies. The 1x1 convolution approach is shown in Figure 3.13.



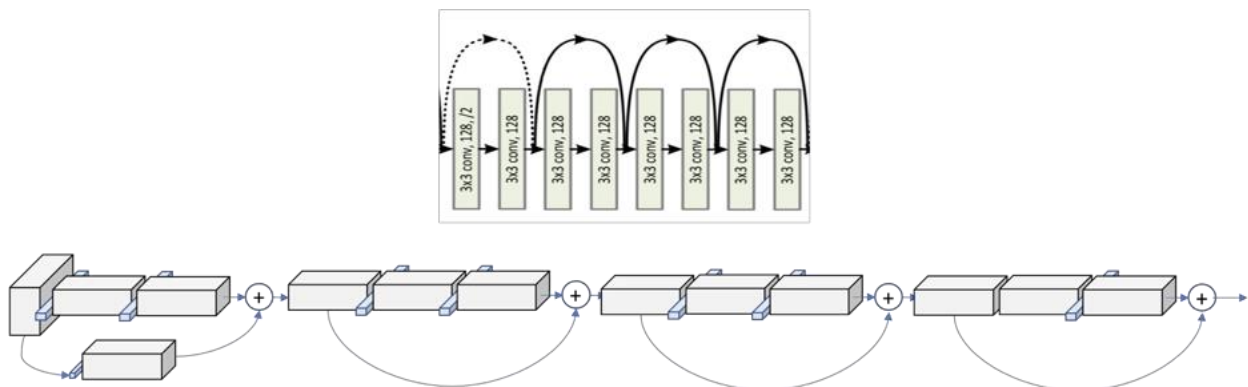
**Fig:3.14: Projection Shortcut Layer 2, Block 1**

The final picture looks then like in Figure 3.14 where now the 2 output volumes of each thread has the same size and can be added.



**Fig:3.15: Layer 2, Block 1**

In Figure 15 we can see the global picture of the entire second layer. The behavior is exactly the same for the following layers 3 and 4, changing only the dimensions of the incoming volumes.



**Fig:3.16: Layer**

## 3.8 Building Blocks of CNN

Convolutional Neural Networks are mainly made up of three types of layers:

**3.8.1 Convolutional Layer** It is the main building block of a CNN. It inputs a feature map or input image consisting of a certain height, width, and channels and transforms it into a new feature map by applying a convolution operation. The transformed feature map consists of different height, width, and channels based on the filter size, padding, and stride.

**3.8.2 Pooling Layer** Pooling layers do the operation of down sampling, where it reduces the size of the input feature map. Two main types of pooling are Max Pooling and Average Pooling.

## 3.9 TENSOR FLOW

**Tensor Flow** is an open-source software library. Tensor Flow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well!

Let us first try to understand what the word Tensor Flow actually mean!

Tensor Flow is basically a software library for numerical computation using data flow graphs where:

- **nodes** in the graph represent mathematical operations.
- **edges** in the graph represent the multidimensional data arrays (called **tensors**) communicated between them.

## 3.10 EPOCHS

Epochs are defined as the total number of iterations for training the machine learning new model with all the training data in one cycle. In Epoch, all training data is used exactly once. Further, in other words, Epoch can also be understood as the total

number of passes an algorithm has completed around the training dataset. A forward and a backward pass together counted as one pass in training.

Usually, when a machine learning model is trained, then it requires a little number of Epochs. An Epoch is often mixed up with iteration. Let's understand the iteration and epoch with an example, where we have 3000 training examples that we are going to use to train a machine learning model.

In the above scenario, we can break up the training dataset into sizeable batches. So let's suppose we have considered the batches of 500 examples in each batch, then it will take 6 iterations to complete 1 Epoch. Mathematically, we can understand it as follows;

- Total number of training examples = 3000;
- Assume each batch size = 500;
- Then the total number of Iterations = Total number of training examples/Individual batch size =  $3000/500$
- Total number of iterations = 6
- And **1 Epoch = 6 Iterations**

### 3.11 DATA SET

We have collected the car data set from Kaggle.

The Cars dataset contains 16,185 images of 196 classes of cars. The data is split into 8,144 training images and 8,041 testing images

Image Source: Towards Data Science

|                      |   |               |
|----------------------|---|---------------|
| DATA set used        | : | "CAR DATA"    |
| Features             | : | Model         |
| Data set format      | : | ".jpg"        |
| No of images trained | : | 16,185 images |
| No of models of car  | : | 196           |
| Each model           | : | 30-40 images  |
| Size of each image   | : | 50-200 kb     |

Our dataset consists of 16,185 total images (train set + test test) labeled with 196 classes based on the car's Make/Model/Year. Here these images come in various sizes and resolutions. For this analysis, the 196 image labels in the dataset were consolidated to five vehicle types as shown in the figure below. Although these represent somewhat 'crude' consolidation of vehicle types, they proved to be more manageable, and adequate for the image classification task.

Features for each of the car images were extracted from Deep Learning Convolutional Neural Networks (CNN) with weights pre-trained on ImageNet dataset. The creators of these CNNs provide these weights freely, and modeling platform Keras provided a one stop access to these network architectures and weights

## CHAPTER 4

### RESULT AND DISCUSSION

Here we have trained our model with resnet34 architecture by training 16,185 images of different models of cars. We have trained this model with GPU (graphical processing unit), it will take one day to train this many images. After that we have done normalizing and reshaping to required size to all images of train, test and validation images. After we set epochs to ten so that it will get trained by back tracing like a loop. After the model get trained, we have tried with different inputs from testing images to test the model. We got 82 percent accuracy to this model. And the accuracy may get increased by 5-10 percent, if epochs is set to 20.

#### 4.1 Training Classes

**(196 classes of different cars, 17000 images in total)**

This is the output of the model after training the 196 model classes, and the total trained images are 16,185 images. Here we are using GPU (graphical processing unit) interface rather than CPU (central processing unit) since CPU will take seven days to train our model as GPU take one day to train.

Each having a 30-40 images and size of 50-200kb

Here we taking data set as “.jpg” format.



**Fig:4.1: Car Data**

### Sample display of 2 images from training dataset



***Fig:4.2: Training Dataset***

This is the sample output to display of 2 images after training the data set. These are some of the training images we have given after the model get trained.

### 4.2 Reshaping, and Normalizing

Reshaping and normalizing the input images for training, testing and validation. Loading dataset images from directory.

```

train_transforms = transforms.Compose([transforms.Resize((244,244)),
                                       transforms.RandomRotation(30),
                                       transforms.RandomHorizontalFlip(),
                                       transforms.ToTensor(),
                                       transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

# The validation set will use the same transform as the test set
test_transforms = transforms.Compose([transforms.Resize((244,244)),
                                       transforms.CenterCrop(224),
                                       transforms.ToTensor(),
                                       transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

validation_transforms = transforms.Compose([transforms.Resize((244,244)),
                                             transforms.CenterCrop(224),
                                             transforms.ToTensor(),
                                             transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

# Load the datasets with ImageFolder
train_data = datasets.ImageFolder(data_dir + '/train', transform=train_transforms)
test_data = datasets.ImageFolder(data_dir + '/test', transform=test_transforms)
valid_data = datasets.ImageFolder(data_dir + '/valid', transform=validation_transforms)

# Using the image datasets and the trainforms, define the dataloaders
# The trainloader will have shuffle=True so that the order of the images do not affect the model
trainloader = torch.utils.data.DataLoader(train_data, batch_size=128, shuffle=True)
testloader = torch.utils.data.DataLoader(test_data, batch_size=32, shuffle=True)
validloader = torch.utils.data.DataLoader(valid_data, batch_size=32, shuffle=True)

```

**Fig:4.3: Reshaping, and Normalizing**

After training the data set we are reshaping and normalising the input images for training, testing and validation. Here we loading dataset images from directory.

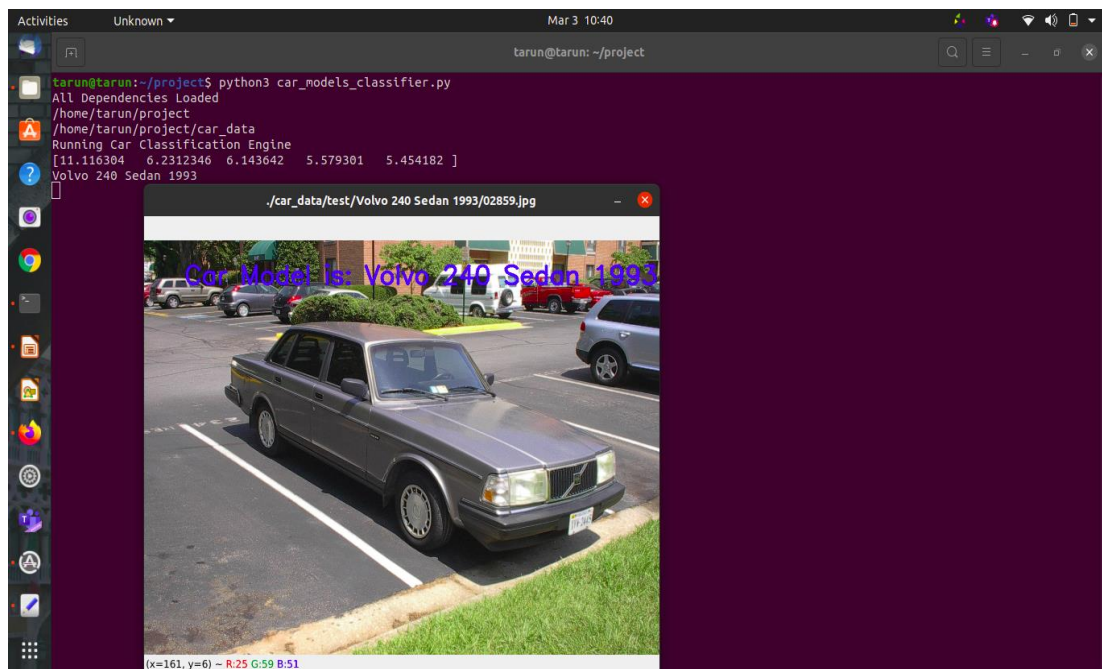
Here, by normalization, we mean changing  $x$  to  $x/\|x\|$  it leads to a better performance because gradient descent converges faster after normalization. And reshaping the image size to required [244,244].

### 4.3 Initializing resnet34 model

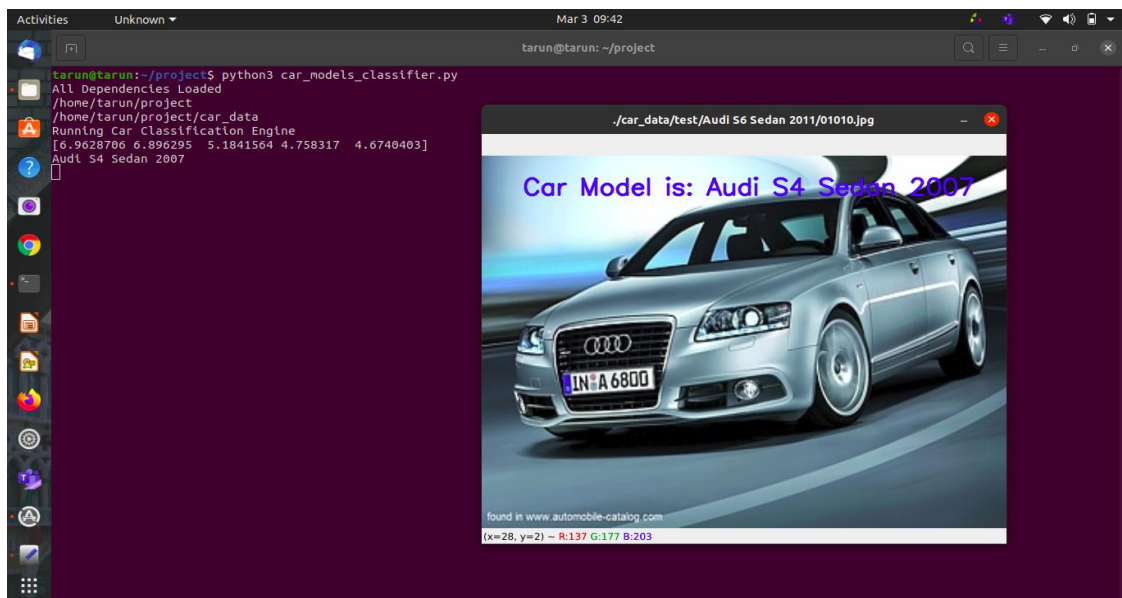
For training and setting cross entropy as loss function. Here in this part we are initializing resnet34 model training and setting cross entropy as loss function. Here we using ResNet-34, since it is a 34-layer convolutional neural network that can be utilized as a state of art image classification model. This model has pre-trained on the ImageNet dataset as the dataset that has 100,000+ images across 200 different classes.

And with large number of layers can be trained easily without increasing the training error percentage. ResNets help in tacking the vanishing gradient problem using identity mapping.

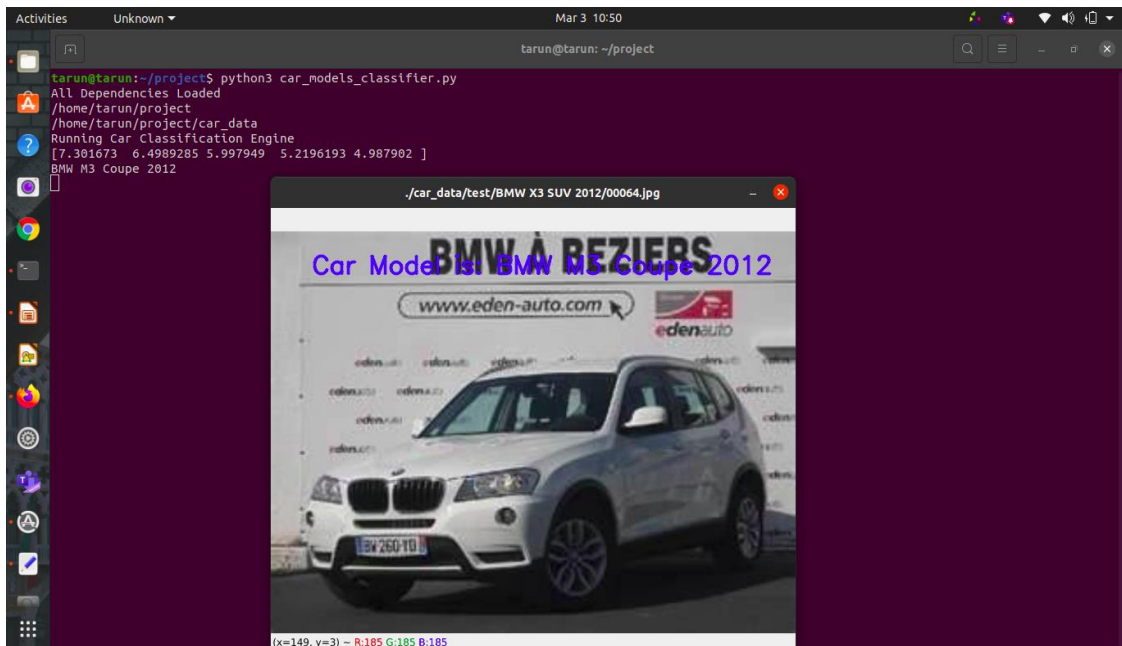
## 4.4 OUT PUT



**Fig:4.4.1: Output 1**



**Fig:4.4.2:Output 2**

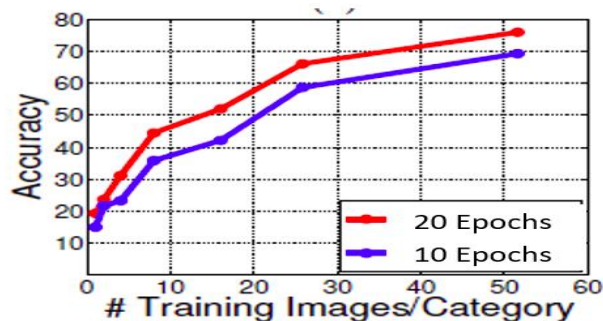


**Fig:4.4.3:Output 3**

This is our final output for this model, here it give the output as image (.jpg format) having model name printed on it. And we printing the directory of the image where it is stored.And we are printing the accuracy array for testing image.

As we see in the above images we hace given different car images as input and output also we have seen like mentioned as car name ,model and year of car manufactured.Since we trained with more images and with efficient data set we got output with more accuracy.

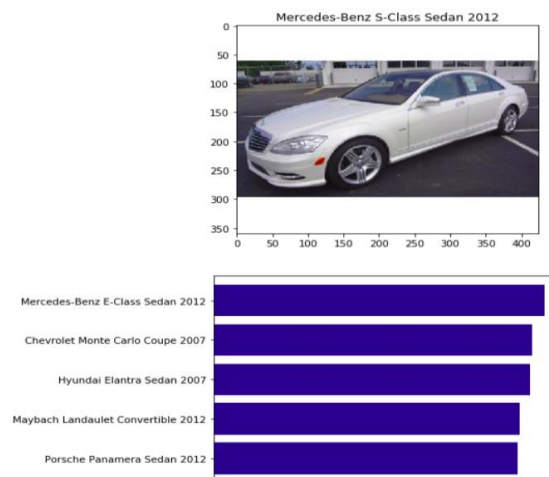
## 4.5 GRAPH



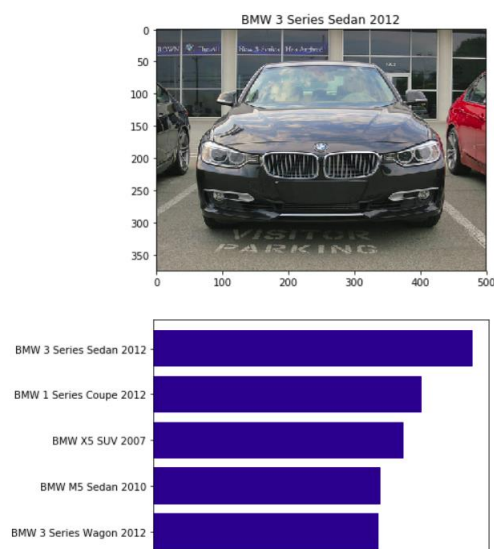
**Fig:4.5: Graph**

This is the epochs graph between accuracy and training images. From this we can say that number epochs will increase the accuracy of the model. For this model we used 10 epochs and we got the average accuracy of 82 percentage and if we use 20 epochs then we may get more than 87 percent.

#### 4.6 MIS PREDECTION OUTPUT

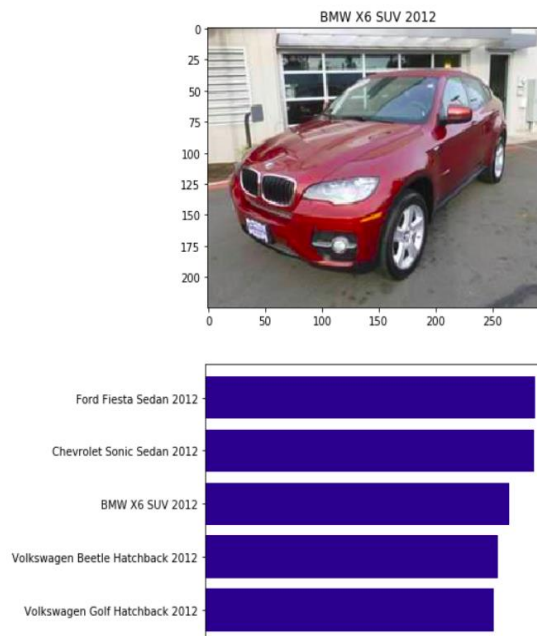


**Fig:4.6.1: Miss Prediction Output**

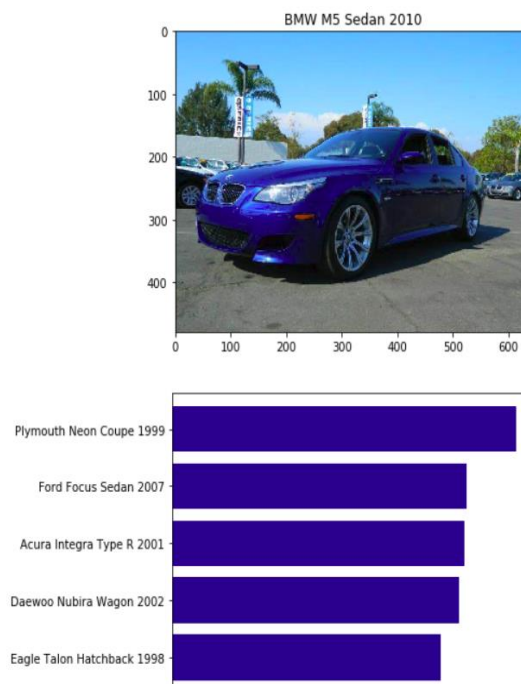


**Fig:4.6.2: Miss Prediction Output**

## 4.7 CONFIDENCE OUTPUT



**Fig:4.7.1: Confidence Output**



**Fig:4.7.2: Confidence Output**

A trained deep learning neural network model can predict only at certain accuracy. We take the ground truth and estimate what is predicted rightly and what is not. And this is called correct prediction and miss-prediction.

## **CHAPTER 5**

### **CONCLUSION AND FUTURE SCOPE**

#### **5.1 CONCLUSION**

Automatic detection of small objects such as vehicles is a very challenging task, due to the complexity of the background, vehicles colors, the large size of ground sample distance. Many methods were proposed for this task by using handcrafted features along with support vector machine classifier, however, Convolutional Neural Networks (CNN) have proved to be potentially more effective.

Training this model with car dataset contains 16,185 images of 196 classes of cars. After we trained this model with 8,144 images 8,041 images for testing. We used GPU interface to train our model rather CPU interface since it takes seven days to train all the images as training with GPU will take one day. After that we are reshaping and normalizing the input images for training, testing and for validation by loading the data set from the directory. Here we write a software pipeline to identify vehicles model using data set. In this implementation, we used a Deep Learning approach to image recognition. Specifically, in this we leveraged the extraordinary power of Convolutional Neural Network (CNNs) to recognize images.

Have trained this type of a model, the input's width and height dimensions can be expanded arbitrarily, transforming the output layer's dimensions from 1x1 to a map with an aspect ratio approximately matching that of a new large input. Here this model classifies and detects whether a vehicle is a car, bus or a bike. This paper creates a convolution neural network from scratch to classify and detect these vehicles using a modern convolution neural network based on fast regions. Finally, car model name and year of manufacture will be displayed on with the image of car.

This model can be used as real time detection on embedded device with high accuracy, and can detection car at time and we can detect color of the car as extension to this project.

## REFERENCES

1. Alon .A, "Tree Extraction of Airborne LiDAR Data Based on Coordinates of Deep Learning Object Detection from Orthophoto over Complex Mangrove Forest", International Journal of Emerging Trends in Engineering Research, vol. 8, no. 5, pp. 2107-2111, 2020
2. Coifman.b, "Vehicle level evaluation of loop detectors and the remote traffic microwave sensor", Journal of transportation engineering, vol. 132, no.3, pp. 213-226, 2006
3. Liu.Y, P. Sun, N. Wergeles and Y. Shang, "A survey and performance evaluation of deep learning methods for small object detection", Expert Systems with Applications, vol. 172, p. 114602, 2021
4. Lowe. D.G., "Distinctive image features from scale invariant key points", International journal of computer vision, vol 60, no 2, pp. 91-110, 2004.
5. Montalbo .F and A. Alon, "Empirical Analysis of a Fine-Tuned Deep Convolutional Model in Classifying and Detecting Malaria Parasites from Blood Smears", KSII Transactions on Internet and Information Systems, vol. 15, no. 1, 2021
6. Mithun N.C, N.U. Rashid, and S.M. Rahman, "Detection and classification of vehicles from video using multiple time-spatial images", IEEE Transactions on Intelligent Transportation Systems, vol 13, no 3, p. 1215-1225, 2012
7. Ritchie.S, and C. Oh, "Real-time traffic measurement from single loop inductive signatures", Transportation Research Record: Journal of the Transportation Research Board, (1804), pp. 98-106, 2002.
8. Rabi.H, "Vehicle detection and classification for cluttered urban intersection", International Journal of Computer Science, Engineering and Applications, vol 3, no 1, p. 37, 2013.
9. Ramakrishnan.V, A. K. Prabhavathy, and J. Devishree, "A survey on vehicle detection techniques in aerial surveillance", International Journal of Computer Applications, vol 55, no 18, 2012.

10. Rocha Filho G, "Enhancing intelligence in traffic management systems to aid in vehicle traffic congestion problems in smart cities", *Ad Hoc Networks*, vol. 107, p. 102265, 2020.
11. Stauffer.C, and W.E.L. Grimson, "Learning patterns of activity using real-time tracking", *IEEE Transactions on pattern analysis and machine intelligence*, 2000. Vol 22, no 8, pp. 747-757, 2000.
12. Seenoung.N, U. Watchareeruetai, C. Nuthong, K. Khongsomboon, "A Computer Vision Based Vehicle Detection and Counting System", *IEEE 8th International conference on Knowledge and Smart Technology (KST)*, pp.224-227, 2016.
13. Shaaban.K, A. Maher and A. Singhapakdi, "Effect of vehicle safety recalls on the perception towards other vehicles from the same country of origin", *Journal of Traffic and Transportation Engineering (English Edition)*, vol. 6, no. 5, pp. 493-503, 2019
14. Yang, "Video Vehicle Detection Based on SelfAdaptive Background Update", *Journal of Njing Institute of Technology (Natural Science Edition)*, vol 2, p. 13, 2012
15. Zivkovic.Z, and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction", *Pattern recognition letters*, vol 27, no 7, pp. 773-780, 2006.