Application Of CNN Based Architectures in Detection of Distracted Drivers

by

Irfana Arafin
22241181
Md Mahirul Islam
18101347
Syed Ittisaf Tazwar
18301137
Nilay Shuvra Das
22241166
Sabrina Tabassum Anika
19301111

A Final thesis Report submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
August 2022

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

[illegible]

Irfana Arafin

22241181

Mahirul Islam

Md Mahirul Islam

18101347

Tazwar

Syed Ittisaf Tazwar

18301137

[illegible]

Nilay Shuvra Das

22241166

Anika

Sabrina Tabassum Anika

19301111

Approval

The thesis/project titled "Distracted Driver Detection Using Deep Learning" submitted by

1. Irfana Arafin (22241181)
2. Md Mahirul Islam (18101347)
3. Syed Ittisaf Tazwar (18301137)
4. Nilay Shuvra Das (22241166)
5. Sabrina Tabassum Anika (19301111)

Of Summer, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on August 23, 2022.

Examining Committee:

Supervisor:
(Member)

[illegible]

Faisal Bin Ashraf

Lecturer
Dept. of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Md. Khalilur Rhaman

Dr. Md. Khalilur Rahman

Associate Professor
Dept. of Computer Science and Engineering
Brac University

Abstract

Distracted driving is known to be one of the most significant reasons behind the occurrence of traffic accidents. Moreover, the phenomenon of the occurrence of road accidents due to distracted driving has been increasing at a high rate in recent years. Previously, different machine learning and neural network-based approaches were taken to find out the best possible way of detecting distracted driving. This work proposes an effective interpretation which is to detect the distraction of drivers through a Deep Learning approach through the implementation of several Convolutional Neural Network (CNN) based architectures. The results presented in this research is to confirm the better accuracy and success rate of the Deep Learning approach to detect distracted driving behaviors demonstrating the potentiality of this method to help measure unusual driving performance. The proposed custom CNN model not only ensures an impressive accuracy but also it's ability to interpret the proper regions of interests on two datasets of distracted driving.

Keywords: Deep Learning; Machine Learning; Distracted Driving; ; Prediction; Decision tree; Linear Regression Analysis

Acknowledgement

All praise to the Great Allah for whom our thesis has been completed without any major interruption.
Firstly, our research is solely dedicated to all the families that suffered from loss of their beloved family member due to occurrence of distraction while driving
Secondly, to our supervisor Faisal Bin Ashraf Sir for his kind support and advice in our work. It is only his guidance that helped and motivated us to successfully present this work.
And finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

Chapter 1

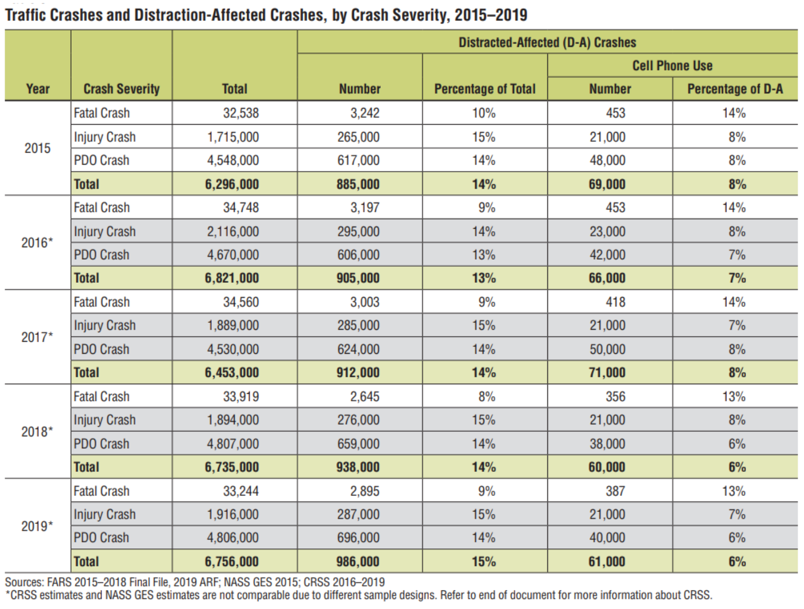
Introduction

1.1 What is Distracted Driving?

Activities that cause distractions by diverting drivers' attention from driving are stated as distracted driving. Such activities may include usage of cell phones while driving, consuming food or drink, communicating with others and so on. It has been proved in several studies that messaging while driving can increase the risk of facing an accident to multiple times.[24]. According to [24], in 2020 the number of fatalities in distraction-affected crashes was 3142 which was 8.1 percent of all fatalities. It claimed 2,628 lives in 2018 and caused 3,003 deaths in 2017. Furthermore, it has been found that approximately 280,000 injuries per year. In 2018 around 276,000 were severely injured, which increased in 2017 by 9000. In 2016, the rate of injuries inclined again and around 295,000 people suffered due to distracted driving[23].
.
Only In the U.S, around 3,477 people died and 391,000 people were severely harmed in 2015 due to distracted driving. [26]Such risky behavior poses danger to both drivers and pedestrians. Distracted driving causes approximately 3,000 deaths per year. In 2019, distraction-affected crashes were reported in 9% of death crashes and 15% of injury crashes, with 6% of the drivers engaged in those fatal crashes being reportedly classified as distracted. [[23],[24]]

1.1.1 Attributes of Distracted Driving

Drivers' distractions may be caused by numerous reasons. However, some significant ones have been identified after several research studies. Here, Figure: 1.1 shows the rise in road accidents caused by the lack of attention of the driver.



Traffic Crashes and Distraction-Affected Crashes, by Crash Severity, 2015–2019

Year	Crash Severity	Total	Distracted-Affected (D-A) Crashes			
					Cell Phone Use	
			Number	Percentage of Total	Number	Percentage of D-A
2015	Fatal Crash	32,538	3,242	10%	453	14%
	Injury Crash	1,715,000	265,000	15%	21,000	8%
	PDO Crash	4,548,000	617,000	14%	48,000	8%
	Total	**6,296,000**	**885,000**	**14%**	**69,000**	**8%**
2016*	Fatal Crash	34,748	3,197	9%	453	14%
	Injury Crash	2,116,000	295,000	14%	23,000	8%
	PDO Crash	4,670,000	606,000	13%	42,000	7%
	Total	**6,821,000**	**905,000**	**13%**	**66,000**	**7%**
2017*	Fatal Crash	34,560	3,003	9%	418	14%
	Injury Crash	1,889,000	285,000	15%	21,000	7%
	PDO Crash	4,530,000	624,000	14%	50,000	8%
	Total	**6,453,000**	**912,000**	**14%**	**71,000**	**8%**
2018*	Fatal Crash	33,919	2,645	8%	356	13%
	Injury Crash	1,894,000	276,000	15%	21,000	8%
	PDO Crash	4,807,000	659,000	14%	38,000	6%
	Total	**6,735,000**	**938,000**	**14%**	**60,000**	**6%**
2019*	Fatal Crash	33,244	2,895	9%	387	13%
	Injury Crash	1,916,000	287,000	15%	21,000	7%
	PDO Crash	4,806,000	696,000	14%	40,000	6%
	Total	**6,756,000**	**986,000**	**15%**	**61,000**	**6%**

Sources: FARS 2015–2018 Final File, 2019 ARF; NASS GES 2015; CRSS 2016–2019
*CRSS estimates and NASS GES estimates are not comparable due to different sample designs. Refer to end of document for more information about CRSS.

Figure 1.1: Distraction Effected Traffic Crashes 2015-2019 [24]

Interruptions while driving can be categorized into three subcategory:

- 1. Visual distractions: Visual distractions denote tasks in which the driver needs to glance away from the road.
- 2. Manual distractions: Manual distraction occurs when drivers need to do tasks using their hand so they need to take a hand off from the steering wheel.
- 3. Cognitive distractions: Tasks that divert the driver's mind can be denoted as Cognitive distractions. [26].

Some of the everyday tasks that were identified to meet all these three subcategories are texting on phone, drinking, glancing behind, talking to passengers, applying makeup, etc. Using mobile devices during driving is one of the most common forms of distracted driving. With each new revolution to innovation in our transportation system and Automobile industry, there is a higher possibility of checking out at a screen and getting distracted by losing concentration from driving.
According to WHO, the usage of cell phones can higher the possibility of conducting an accident caused by distraction up to four times.[18] National Occupant Protection Use Survey (NOPUS) observes that usage of electronic devices while driving has a higher possibility of leading to accidents. The availability of smart gadgets such as cell phones, and radios have introduced us to additional factors that induce significant threats of road crashes led by distraction [2].

1.1.2 Limitations of Detecting Distracted Driving

The focus of this dissertation is appropriately assessing distraction while driving which has been proven to be very crucial. There are some segments that need to be maintained while driving a vehicle. Those are, physiological parameters, vehicle driving state, and vision.[16] A driving behavior detection system must meet two conditions. First, regardless of the detecting method used, measurement devices should have no effect on the drivers. Although physiological parameter techniques can reach great precision, such measurements need a large number of sensors, which may cause the driver to get distracted. Secondly, lag occurrences of present driving behavior monitoring technologies are a matter of concern as well.

Nowadays, there are two types of detection methods used for distracted driving detection such as traditional detection techniques and Deep Learning based detection techniques. After reviewing some articles and papers, we found that these methods have some downsides. Traditional technology is not capable of distinguishing more than one distracted driving behavior at once. For example, Eating and talking over the phone at the same time can not be detected at once. Thus, the two get identified individually which effects the accuracy severely. To identify drivers' behaviors where they might be using cellular devices while driving, there are monitoring methods where signals coming from a phone gets tracked continually. However, The signal monitoring method also involves putting an antenna in a fixed place so that the signals coming from a phone of that certain vehicle can be monitored even if the vehicle is in a moving state. By using a directional antenna on the road at a fixed position, from the signal source, it determines whether the mobile phone of the moving vehicle is in an active state or not. [16] However in this process, the chances of getting a false detection result is very high because of the detection complexity. The reason is, this process only detects whether a phone is in call state inside a vehicle or not. It's unable to distinguish that the user of the phone is a driver or passenger or both. There is another way to solve this problem but it will be a lengthy process. The detection method mentioned above has a complex structure, is slow to detect, and cannot be employed in practice.[16] Other methods that are based on deep learning for distracted driving detection are the most popular and also the most used. But it also has some problems. Like it needs powerful computing units to train.

1.1.3 Use of Deep Learning in Distracted Driving Detection

The use of Deep Learning in vision related tasks is increasing day by day as Deep learning techniques can improve the accuracy significantly. Because of technological progress of recent years it is possible for real-time algorithms to be able to accurately detect the activities that are causing distraction and also assist to alert the driver to avoid accidents. Alongside, several improved deep learning classification techniques and detection algorithms can be used after breaking down the problem into several smaller segments. In several researches, Convolutional Neural Network(CNN) were used in such way which ultimately helped to improve the accuracy. Also, improved richer datasets can make DL algorithms achieve better accuracy in many terms. [22]

1.2 Problem Statement

The focus of this dissertation is appropriately analyzing drivers' abnormal behavior by detecting manual distraction using Deep Learning based approaches. A webcam can be installed above the dashboard of cars with the purpose of recording the movements of the Driver which later on can extract RGB frames. A combination of Deep Learning models will be developed as the candidate detector after using those RGB-channel frames as inputs. We intend on presenting a real-time system to be implemented on smart vehicles to be able to recognize and distinguish such interruptions. This system can be employed to warn drivers ahead of time to prevent accidents by identifying distractions. In Figure 1.2 the formal representation of our basic model has been presented.

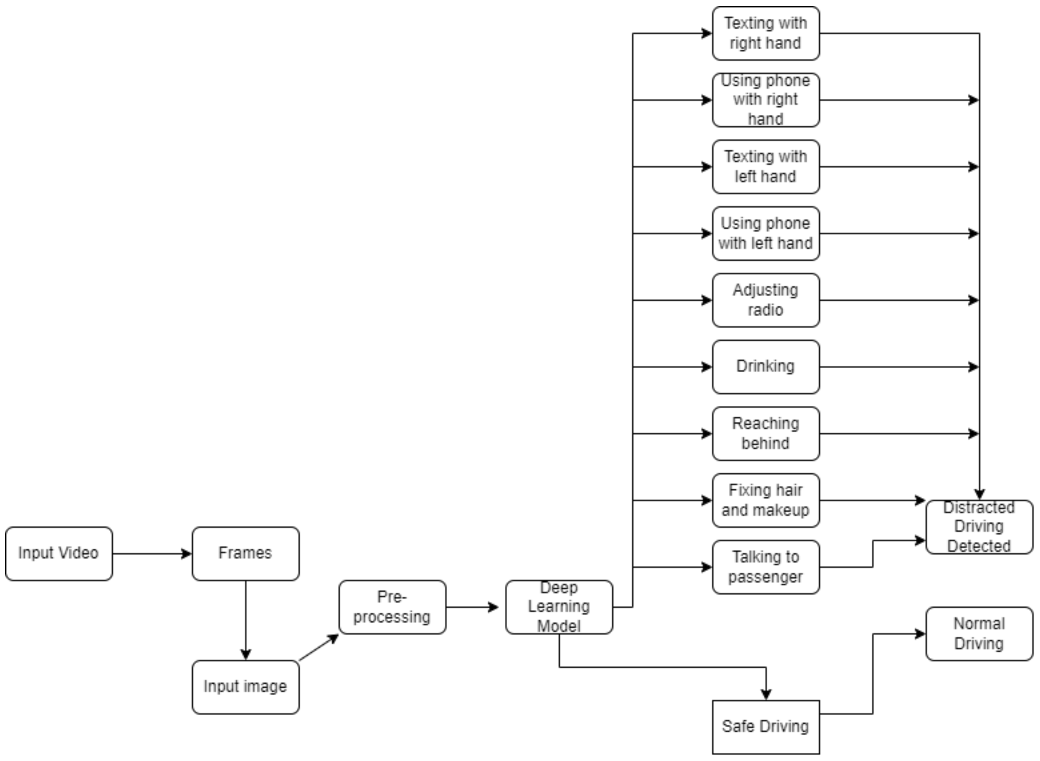


Figure 1.2: Basic representation of the proposed model

It depicts a high-level view of the approach suggested in this study. The goal is to demonstrate Deep Learning's capacity to detect Distracted Driving after being trained with a large enough dataset.
In the Automotive industry, a significant amount of investment is made annually in the mechanization of vehicles so that driving a vehicle feels more secure and more efficient to everyone. Automated driving needs to ensure effortless adaptation between the automation system and the driver. Therefore, in such terms, distraction identification may play a role as an important feature to ensure a smooth transition between the driver and system [20]. Then again, such discovery frameworks can help regulation authorization to recognize interruptions on highway utilizing cameras. We envision ensuring proper detection of Manual, Cognitive and Visual distractions of drivers to provide a more reliable transportation experience for everyone which will ultimately help to avoid unwanted road tragedy.
So far, different approaches have been made to enable accurate distraction identification. Mass use of Convolution Neural Network(CNN) has been made to recognize distracted behaviors. Research that has been made over the last seven years, in the field of distracted driver detection has been categorized into four categories [13]. Those are Cell Phone Usage Detection where they offer an SVM-based model for detecting cell phone use while driving. A front visual image of a driver's face is featured in their dataset. Secondly,in UCSD's LISA works a vision-based evaluation

scheme has been described that requires two Kinect cameras to detect drivers' activity from front and back.[1] Their method is to extract information from wheel handling scheme to detect three different forms of disruptions. Thirdly, the Southeast University Distracted Driver Dataset includes distracted driving dataset with front visual image of a driver's body which can cover his side face and hand and also the driving seat. This way, additional actions such as whether the driver is gripping the steering wheel, shifting gears, eating, and chatting on a cell phone can be captured by a camera. [1] It presents the contourlet transform for feature extraction which later on is compared to the performance of Random Forests (RF), K-Nearest Neighbors (KNN), and Multilayer Perceptron (MLP) classifiers. Lastly, vast use of StateFarm's Distracted Driver Detection Dataset was seen in detection of distracted driving. This dataset has been the first publicly accessible dataset on Kaggle for distraction classification where they identified 10 major activities that causes distraction. Those formed 10 different classes to be distinguished as distracted driving [13].

In [14], by operating a combination of both Deep Learning and Computer Vision, the system was able to identify drivers' drowsiness while driving. The system rang the caution bell whenever the driver was detected to be drowsy(Figure 1.3), and then image recognition was completed using an enhanced version of basic Convolutional Neural Network model. It was used to improvise the recognition of the location of the face and eyes in a proper manner in order to improve the detection results while lowering the percentage of false positives and negatives.

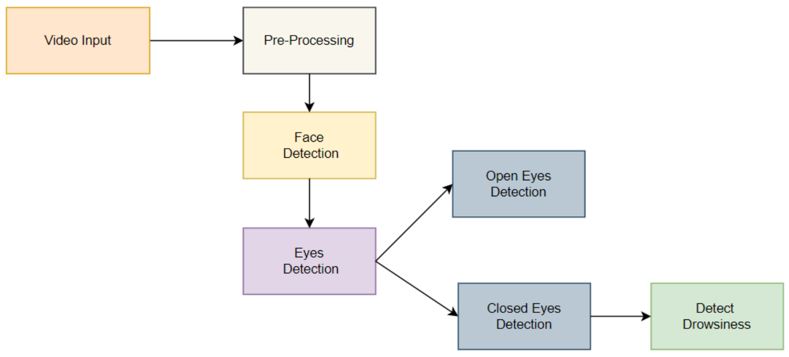


Figure 1.3: The traditional way for detecting driver drowsiness [14]

If the facial gesture identification feature is added to our previously introduced basic model(Figure-1.2), then the driver's facial landmarks, as well as the movement of his eyes and mouth can be continually tracked using a camera. In that case, we will be able to detect all Manual and Visual distractions. The advanced version of our basic proposed model is presented in Figure 1.4.

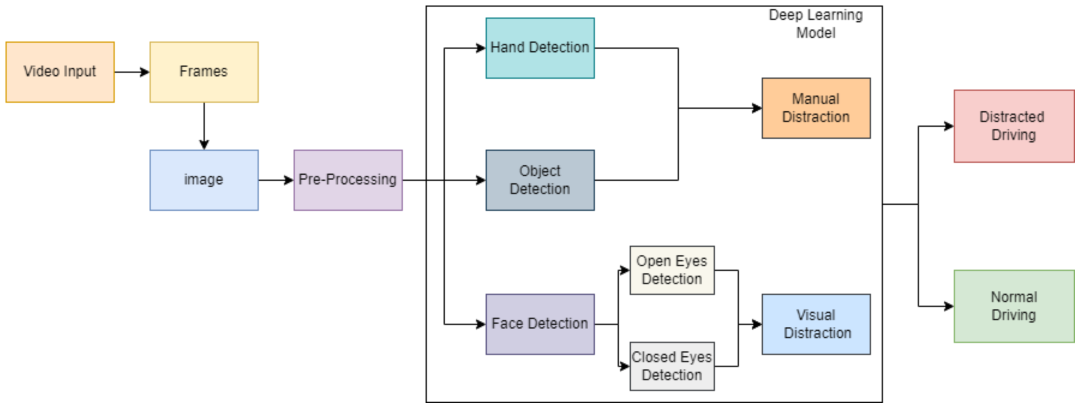


Figure 1.4: Overview of Proposed Model

1.3 Research Objectives

A conventional technique must be established following standard procedures to be able to accurately detect distractions while driving using Deep Learning Models. Data from numerous sources should be merged into driver distraction detection systems constructing a cooperative sensing system to synchronize data sets, connect relevant features and combine them to detect or perform classification which is one method to fulfill this requirement [5]. Through the use of our desired Deep Learning models, we intend to be able to minimize computing complexity while maintaining high accuracy which is indeed needed in real-time-based implementation.

We seek our Deep Learning Models to achieve significantly higher performance and accuracy by detecting both Manual and Visual distractions while in most research only one segment was seen to get prioritized in terms of Distracted Driving classification. To gain such state-of-art accuracy our extracted image needs to be fed into merged combination state DL models.

1.4 Paper Orientation

Chapter 1, primarily introduces us to the definition of Distracted Driving and its significance in our livelihood. Several essential factors for the occurrence of driver distraction detection are discussed in this section, particularly distracted indications and its attributes, some basic classification techniques, and individual variances. The characteristics of the type of distraction have an impact on the selection of these attributes. Therefore, Chapter 2 summarizes the topics that needed to be learned to perform the necessary implementations in this work. Whereas, Chapter 3 provides literature review of researches that have been made in Distracted Driving Detection. Chapter 4 gives an describes the dataset used in this research. Hence the models that have been implemented in the dataset and their architecture has been described in Chapter 5. Chapter 6 demonstrates the results obtained after implementation of those models. Chapter 7 describes the probable causes of obtained performance from the proposed model and Chapter 8 concludes while summarizing the entire work and it's potential. Finally, there is a bibliography at the conclusion that identifies all of the sources and publications that were mentioned in this paper.

Chapter 2

Background

2.1 Convolutional Neural Network

Convolutional Neural Networks have delivered leading-edge findings in the spectrum of pattern recognition-related domains, including speech recognition and image processing. It is a Deep Learning method which takes image inputs in terms of tasks related to image processing while assigning distinct aspects and objects in the image, such as, weights and biases which allows it to distinguish different aspects or features among those images after several processing. In this manner, every layer of a CNN learns several features, which are basically numerous sets of weights connecting it to the former layer. An image merely consists of a matrix of pixel values, and therefore the mathematical process of Convolution involves combining two primary functions to generate a third one.

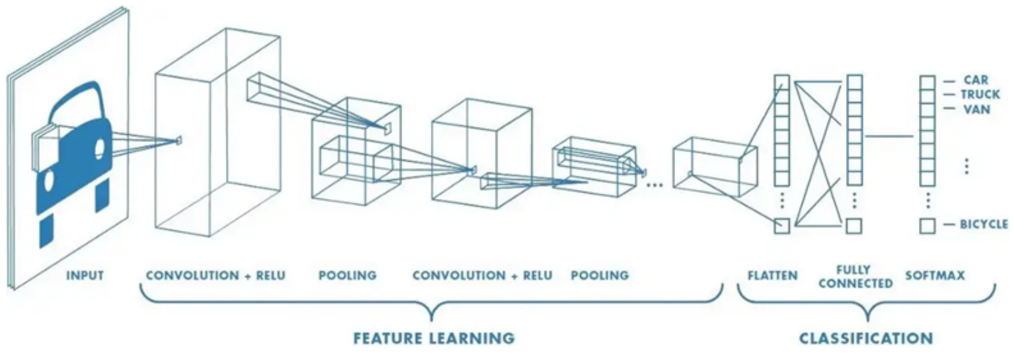


Figure 2.1: Basic Convolutional Layer Architecture

A Convolutional neural network typically consists of three prime layers which are a convolutional layer, a pooling layer, and a fully connected layer, also known as dense layers. Traditional neural networks work more like fully connected layers where all nodes are connected which ultimately results in a massive increase of parameters and computational complexity increases too. CNN introduces convolution layers with non-linearity and pooling layers due to which this issue reduces drastically. The architecture of the model gets flexible too. Some of the popular CNN architectures are LetNet, AlexNet.

2.2 Foundational Elements of CNN

2.2.1 Input Layer

This layer contains the input image which can be a 3D RGB image that is a three channeled image made up of the colors red, green, and blue. It may also be in a grayscale form, which is single-layer two - dimensional image. The construction of the anticipated kernel shapes reveals the primary distinction between the input arrangements. When the images are in 3-channeled form or RGB, the input dimension is N x N x 3, where N resembles image size. In this work, the image dimension has been kept as, 224 x 224 for all models that has been implemented, while implementing in RGB form, resulting in an input shape of 224 x 224 x 3.

2.2.2 Convolution Layer & Kernel

When the neural network receives raw pixel inputs, adding more neurons in hidden layers may increase the parameter amounts drastically. To drop the size of connection instead of using fully connected layers it is possible to look into local regions of the image instead of the whole. Whereas, to reduce parameters we can keep the local connection weights fixed for the entire neurons of the next layer. This way weights will remain same for next chosen layers. Therefore, it again drops many extra parameters, and reduces the number of weights. Convolution layers give us the opportunity to detect and recognize features regardless of their positions in the image. A kernel is a matrix of values that is processed by the input in order to retrieve key information.

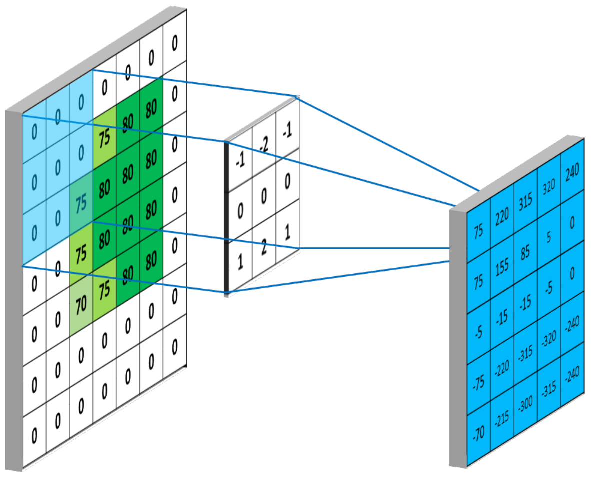


Figure 2.2: One of the stages of the convolution process.

Convolutions can be of one-dimension, two-dimension, or three-dimensions. Two-dimensional matrix kernels are used by 2D convolution layers. There are several numbers of nodes that all focus on the same region of the picture, therefore learning more than just one 2D Kernel's weights is necessary. Each node in row performs attempts to understand several kernels that are composed of various weights which will eventually reveal various characteristics of the image.

In the Figure above, a 3 x 3 kernel that has different weights assigned to it, is applied on an image with 2 x 2 zero padding. Then a maxpool operation is performed with stride of 1 resulting in an output image that has not shrunk. Whereas, a filter, on the other hand, is a combination of many kernels, each paired with a certain input channel. The difference between filters and kernels is, filters have one dimension more than of kernels. For instance, 2D convolutions use 2D kernels and filters will be 3D.

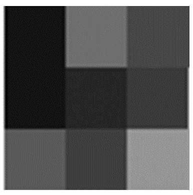


Figure 2.3: One of the stages of the convolution process.

2.2.3 Activation Function

The activation function of last layer solely depends on the task of the model. For example, in terms of binary classification there are some activation functions and for multiclass classification there are different sort of activation functions. For binary classification and multiclass multi-class classification usually Sigmoid is used. For Multiclass- single class classification we use softmax. For our problem Softmax has been used as an activation function.

2.2.4 Strides

The space between two consecutive kernel positions is termed as a stride which is 1 by default. However, occasionally the feature maps are down sampled using a bigger kernel size. Stride is used as one of the methods to reduce parameters of CNN architecture. In CNN, each layer is associated with filters, thus each layer extracts different features from the input. Meanwhile, there are chances that while looking at the regions of the input, there are overlaps among multiple layer's nodes. Those overlaps can be manipulated by the use of strides in convolution layers. Through strides not only issues of overlapping can be solved but also the size of output may be reduced.

If stride is implemented, the output shape of the feature can be calculated with equation 2.1.

$$O = 1 + \tfrac{N-F}{S} \tag{2.1}$$

Here, O resembles the output size, N denotes to image dimension, Filter size is F and S points to number of strides implemented.

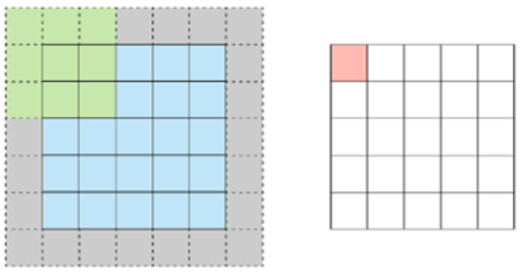


Figure 2.4: Stride 1 With Padding & Resulted Feature Map.

2.2.5 Padding

Due to the convolution step, information that may exist in the very edge of the image, may never be attained as this is gained every time the filter slides. This way there are always chances of that information getting ignored through each convolution step. To solve this issue, zero padding has been proposed. Due to stride even though the input shrinks, it is possible to keep information intact by the use of zero padding.

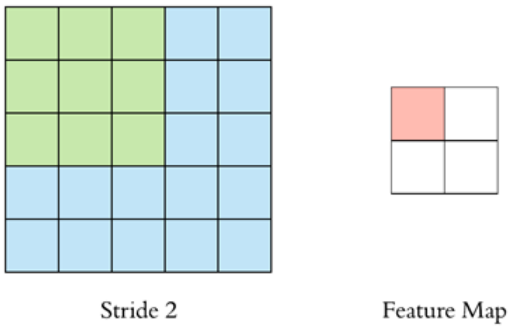


Figure 2.5: Stride 2 Without Padding In Feature Map Of Fig X.

If padding is implied, the output equation will be,

$$O = 1 + \frac{N+2P-F}{S} \tag{2.2}$$

Where P represents the number of zero padding that has been used in the convolution layer.

2.2.6 Pooling Layer

In order to down sample feature map size in order to establish a translation invariance that identifies minor shifts and deformation and lessens the number of learned parameters, the pooling layer is used. Through max-pooling it is possible to reduce the complexity for further layers. Convolutional neural networks usually show impressive results in deep learning, particularly in the image-processing domain. Max-pooling divides the image into rectangular sub-regions and only delivers the maximum value of that sub-region, with no impact on the number of filters. The most common max pooling is stride of 2 and filter size of 2x2. When the pooling is performed in the red 2×2 blocks it moves 2 steps each time while focusing on the image's green part. This means that stride 2 is used in pooling. As we see in Figure Y, the feature map is divided within patch sets from which the greatest value from each patch is chosen and the remaining get eliminated and max pooling is preferred for this pooling operation.

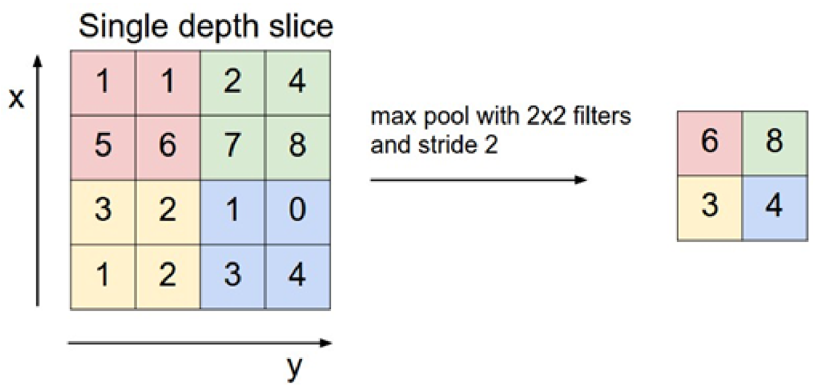


Figure 2.6: Stride 2 without Padding In Feature map of Fig X.

On the other hand, average pooling is when in order to construct a down sampled feature map, the average value across regions of a feature map is calculated. Figure 2.7 below showing the feature map resulting after being implementation of 2x2 maxpool and 2x2 average pool. The difference of calculation among max pooling and average pooling can be seen here.

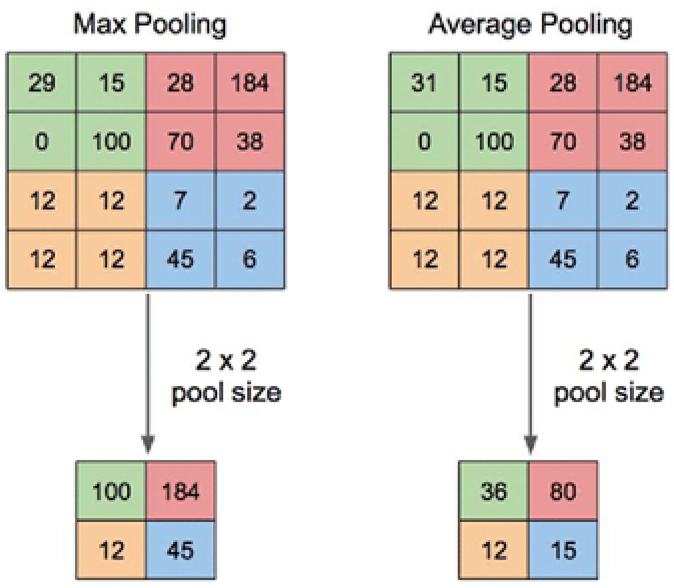


Figure 2.7: Avg-Pooling Vs Max-Pooling.

2.2.7 Dense Layers

The convolutional layer and pooling layers outputs have connections with one or multiple fully connected layers and in this way each input and output has connection by a weight that is learnable. The characteristics need to be retrieved by the convolution layer before the pooling layer down sampled it once, then they generate final outputs similar to probabilities for classification after being forwarded to the layers that are fully connected. The amount of nodes in this layer is dependent on the quantity of classes. As stated above, following each layer comes a nonlinear function, such as ReLU.

2.3 Training Model

The sole purpose to train a model is to achieve a blend of distinctive kernels in the convolution layer and weights for the layers that are fully connected which will generate end products that differ as little as possible from the labeled dataset used. The neural networks that have concealed layers and loss function and gradient descent optimization are trained and back-propagation is the primary approach for that.

2.3.1 Loss Functions

The operation that estimates actual output and the network output difference with the help of forward propagation and marks it as the cost, is called the loss function. Cross-entropy for categorization into many classes and the regression's mean squared error to continuous value are the most commonly used loss function. There are several types of loss functions. Binary loss function is used when the model needs to classify among two classes. In that case, often Binary cross entropy is used as a loss function. However, for multi-class problems such as ours, we need to use loss functions that work in multi-class situations. In our research, we used a categorical loss function as we have 10 classes.

2.3.2 Optimizer: Adam

Adam is a highly advanced adaptive learning rate optimization technique for deep neural network training which is a combination of two optimization algorithms, Root Mean Square Propagation and SGD with momentum [6]. To identify different learning rates for every parameter, its algorithms make use of adaptive learning rate approaches which work impressively in cases of sparse gradients. Adam was developed with the goal of integrating the strengths of AdaGrad. For each weight in the neural network, Adam adjusts the learning rate using estimates of the first and second moments of the gradient [6].

2.4 Splitting Dataset

The most important area whether it's of machine learning or it's of deep learning methods is the sets of data and the ground truth labels. As a matter of fact, any such approaches or models must meet these requirements to be effective. There are several image sources regarding distracted driving that are publicly available. But to become useful to a specific cause, it requires data sets with certain ground truth labels and requires careful attention. Training, validation and test are the 3 categories of datasets. Loss values are estimated using forward propagation and learned parameters in the training set and then use backward propagation to update it back to the network. The fine-tuning of hyper-parameters and model selection is done throughout the training process using the validation set. From figure 8, the completed model or network is executed on a test set and the final result is evaluated at the very end. Here, we can notice that the evaluation sets are kept different from the test sets because the fine-tuning of the hyper-parameters of the training model is done based on its performance on the evaluation set.

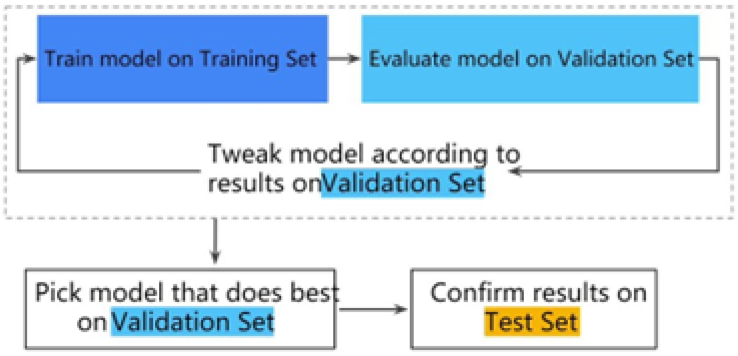

Train model on Training Set
Evaluate model on Validation Set
Tweak model according to results onValidation Set
Pick model that does best on Validation Set
Confirm results on Test Set

Figure 2.8: Division of Dataset in Train, Test and Validation Subsets

2.5 Overfitting & Underfitting

When a function is too accurately matched to its particular data points, it is a case of overfitting which is a common phenomenon. Overfitting occurs when the model has a high variance, indicating that besides learning about the vital information regarding training data, it also learns the irrelevant data or noise specific to the dataset. As a result, even though the model is seen performing great while training the data it may not do as great while evaluating the model with unseen data and performs inaccurate predictions[12]. There are some certain ways to avoid overfitting issues while training data. For instance, as mentioned earlier, small amounts of data on training sets may lead to overfitting and to avoid that, the training data size can be increased. Another reason for overfitting is, proceeding training with imbalance data, which refers to training the model with different classes where the amount of data to be trained for each class are significantly unequal. Vice versa to the situation of less training data, a model's architecture should be constructed or selected considering the data that needs to be dealt with. Moreover, in terms of CNN dropouts of a certain percentage may also be specified to reduce overfitting. Dropout is a regularization strategy whereby randomly chosen activations are set to 0 during the training to minimize the sensitivity of the model to particular weights[3].

Therefore, underfitting is an issue when the model produces poor results in both training and testing phases. Underfitting occurs when a model cannot learn sufficient information from the training data, which lowers performance and results in incorrect predictions. This concept is quite opposite to overfitting. Similar to overfitting there may not be a single reason behind a model being underfitted. Underfitting may occur due to several factors. For instance, if the constructed architecture of a model is too simple compared to the provided dataset, then the model is unable to

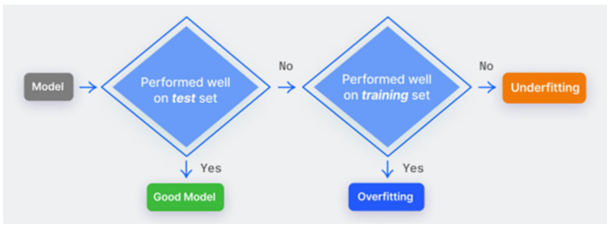


Figure 2.9: Underfitting And Overfitting Flowchart

learn from those data points. As a solution to this, the model should be constructed depending on the provided data. Another solution is to set an early stopping, which may not be feasible at times as it leaves the possibility of the model ignoring later epochs where it might have achieved better accuracy and learning.

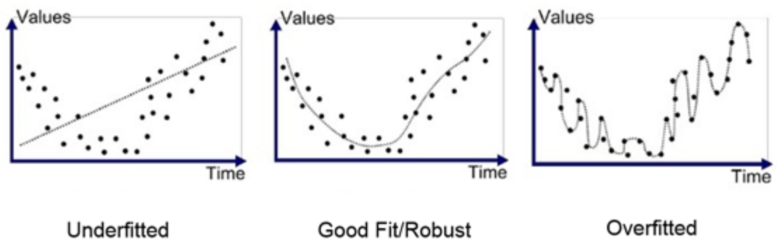


Figure 2.10: Graphs of Overfitting Vs Underfitting Vs Robus Situation

In our work, all of the models have been trained using the training and validation portion of the dataset and the model has been evaluated using the test portion. To avoid any probable chances of overfitting, it has been ensured that every class of the dataset has been balanced. Along with that, dropout has been as well which will be broadly discussed in the Chapter-5.

Chapter 3

Literature Review

Distracted driving is a huge problem in recent times. It causes. In the past, we used to identify distracted drivers by noticing their inability to maintain lane position, sudden swerving for no apparent reason, not being able to keep a relatively constant speed, sudden breaking in reaction to normal traffic stops, or ignoring traffic signs. But these methods are not that effective compared to convolutional neural networks.

3.1 Research Using Several ML Models

ResNet50, Inception-V3, Xception are the pre-trained models used in this paper [22]. They were trained in the dataset (ImageNet). After that, transfer learning was employed to fine-tune feature concatenation modules that are used to deeply fuse retrieved features in cooperative Convolutional Neural Network(CNN). Features of vector's weights were trained by the feature classification.

The paper [11] focuses on resolving the problem of overfitting. There are several methodologies like CNN, VGG16, VGG19, and InceptionV3. In the case of small CNN, the training result is substantially more than the validation result and that causes overfitting. Adding a large number of layers can solve this problem but then the computational cost would be very high. Additionally, utilizing an ensemble of several models rather than a single model will produce improved results. So, instead of using one single model, they use 3 models. The final prediction result comes from averaging these predictions of these 3 models.

Another dissertation [9] describes a three-step procedure. The first one is a specification for a road segment. It forecasts the driver's ability to maintain the center line and speed limit. For each driver, the predictor is trained without secondary activity and training data is gathered. The anticipated driver performance is then compared to the secondary task performance, and the discrepancies between the two are determined. Finally, the Federated Learning (FL) evaluator normalizes two independent variables using linguistic principles to create a uniform variable Distracted Driving (DD), which identifies the DD level in percentage.

This paper [10] discusses procedures such as the original VGG16 architecture and improved VGG16 architecture. To avoid overfitting the training data, they alter the VGG-16 architecture for this job and utilize numerous regularization strategies.

The proposed method beats existing techniques of distracted driver identification in the literature on this dataset, with an accuracy of 96.31 percent.

The VGG-16 architectural model is used in the following paper [17] which always uses 3x3 filters with the same padding in pooling layers 2x2 with a stride of two. Softmax is used as a dense layer activation, while Stochastic Gradient Descent (SGD) is used as a network. Object identification, fine-grained classifications, facial traits, and localization are all common uses for Mobilenet. Transfer learning is another approach employed in this paper. Furthermore, it is a machine learning strategy that focuses on saving info obtained from clearing up an issue and using that knowledge to another problem.[13]

Now we see the use of the EfficientDet model for detection purposes [22]. Picture frames are taken from video and translated into textual labels associated with image categorization and detection categories. With five variations, the EfficientDet model is employed for detection. To develop actual predictions and obtain state-of-the-art outcomes, the model is set for recognizing the items, ROI parts participating in the distracting tasks. It should be noted that five different EffecientDet versions were employed in all. Furthermore, three detection models were used to train the dataset: EfficientDet, Yolo-V3, Faster Region-based Convolutional Network (R-CNN). Here EfficientDet has the highest Mean Average Precision (MAP), as we can see.

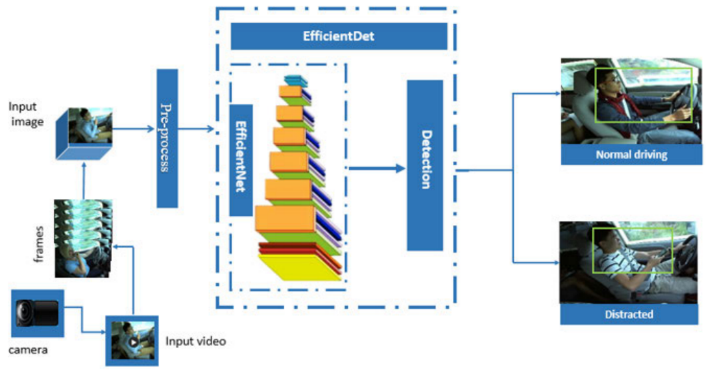


Figure 3.1: Proposed Model of [22]

The proposed solution in this paper[22] proposes an ensemble of CNN as a solution. Raw photos (five picture sources) were used to train the convolutional neural networks. Alex Net, InceptionV3, a ResNet, and a VGG-16 were trained using those five picture sources. For such networks, they fine-tune an ImageNet model called transfer learning. Later on, a genetic algorithm is used to assess the total of all networks' outputs, then provide the final class position.
Each frame required a detector for face, hand and skin segmentation. Networks that have been trained for every output image of skin, face and hands were AlexNet and an InceptionV3. (i.e., results into 10 neural networks where each of these networks has 5). By predicting the weighted sum of the softmax layers, the class distribution was predicted. And, Genetic Algorithm was used to learn the weights. [13]

In another paper[25] three well-known CNN models have been compared to tra-

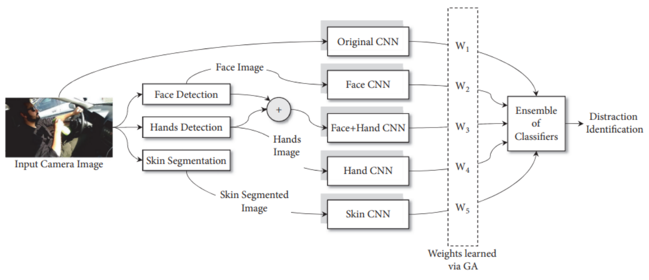


Figure 3.2: Proposed solution in [13]

ditional handcrafted features for automatically detecting whether the drivers are engaging in distracting behaviors or not based on images from a dashboard camera. Using the Softmax layer as a classifier, ResNet-152 produced the best accuracy of 85%, which is higher than VGG-16 accuracy and AlexNet. In addition, using SVM classifiers on the extracted features for the last layers of the CNN models did not increase the accuracy.

Except for mobile phones and the category of external distractions, this study[24] lists sources of distraction. NHTSA and its data consumers, according to its methodology, must be aware of the disparities in definitions of distraction and inherent limits in data collection for distraction-affected collisions, as well as the con- sequent injuries and deaths. This document's appendix offers a table that defines the coding for distraction-affected accidents for FARS and GES, as well as a de- scription of the distracted driving data's limitations.

In another paper [25] shows how HSDDD, which is built on a three-tiered design, works. The first tier does feature extraction. The second tier involves feature concatenation. In the third layer the second layer will provide an output that will be used as an input. After completing training, distracted driving behavior is categorized by several KNN and SVM variations.

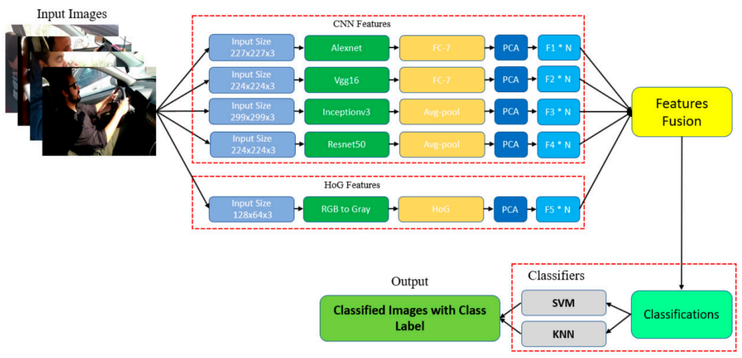


Figure 3.3: An overview of the proposed solution in [25]

The following is the structure of the paper [4] the definition of driver distraction is offered in Section 2 of the article, based on current literature discussion. The third section will briefly cover the machine learning approaches used to model a driver distraction. In Section 4, the experimental setup will be shown. Meanwhile, Section

5 summarizes the key findings. Finally, Section 6 tries to critically evaluate these findings by comparing them to the most important findings of previous studies in this field, highlighting differences, innovations, shortcomings, and potential future actions

This paper[4] uses deep learning and computer vision to identify driving tiredness. The system uses a camera to capture photos of the driver. After performing digital image processing on the collected picture, the driver's conduct was examined and inferred. The technology was split into two parts. One is where it detects if the driver is nodding, and another is sleepiness detection. While performing detection that if the driver wore sunglasses then that was not being detected. Avoiding a scenario was the aim of nodding detection. However, the method of Sleepiness/Drowsiness detection was to assess the duration of the driver keeping his eyelids closed. The driving system uses a variety of sensors to identify and combine multilayer sensations. The CNN model was used to accomplish picture recognition. The advanced CNN model was used to estimate the location of the face and eyes since eye prediction, face prediction, and hand prediction are the three main components of the driving behavior detection system. This improved detection results while reducing false positive and negative rates. Installing a camera on the dashboard also uses YOLO-based deep learning technologies to identify the behavior of the driver. YOLOv3-tiny 3l was an enhanced version of YOLOv3-tiny that was designed to improve small object recognition. In an 8:1:1 ratio, the categorized data sets are separated in training, validation, and testing sets.

This paper [21] suggests utilizing deep learning to identify driver attention, but it will focus on real-time deployment. It will accomplish this by first training a deep learning model on a local PC before delivering it to embedded devices. In addition, real-time metrics will be measured and improved. To make a real-time deployment easier, the feature for detecting distraction must be kept simple, hence this study will employ a driver pose as its feature.

This paper [27], shows a Long Short-Term Memory(LSTM)-based (DBRPNN) which has been introduced to identify driving distraction through prediction. Time attributes and vehicle at- tributes were added together to get the final accuracy. The Neural Network model that was seen in this work predicts the risk of distraction. FPM was used in classification of different features where the neural network also utilizes the risk factor to identify the worst possible extent of upcoming danger. That output is fed into the Memory Module that gets extended to the phase where use of LSTM takes place in this segment because basic Recurrent Neural Network (RNN) falls behind in achieving accuracy in this task. At last, PM is utilized in successfully calculating and converting the output of LSTM to Possible Risk factor. The results show that DBRPNN has an uplifted accuracy and it has large prospects.

In this paper[19] their key element of the suggested framework is the segmenting of the parts of the human body. It's used on the raw RGB image. Because of that the irrelevant objects get removed easily and effectively and also detects the necessary body parts of the driver. After that, the classification model receives the resultant image. Here, firstly the segmentation steps of the human body parts are discussed, and then there's a description of the classification models and the training models that were used to complete the classification of these images.

Here,[28] a new method has been proposed in order to identify the following issues. They come up with a brand-new network architecture that features the Conv and Transformer block. The Conv block was utilized to retrieve local features in the image and for gathering the global information of the image transformer was used. The ResNet and the Vision Transformer(ViT) structures are referred to in this technique. Whereas this transformer block's architecture corresponds towards the ViT, CNN's architecture uses the ResNet network architecture. The ViT approach divides each picture as a patch which includes location embedding to create a series of tokens. With this technique, vectors with parameters are extracted and given pictorial imagery using cascaded transformation blocks.

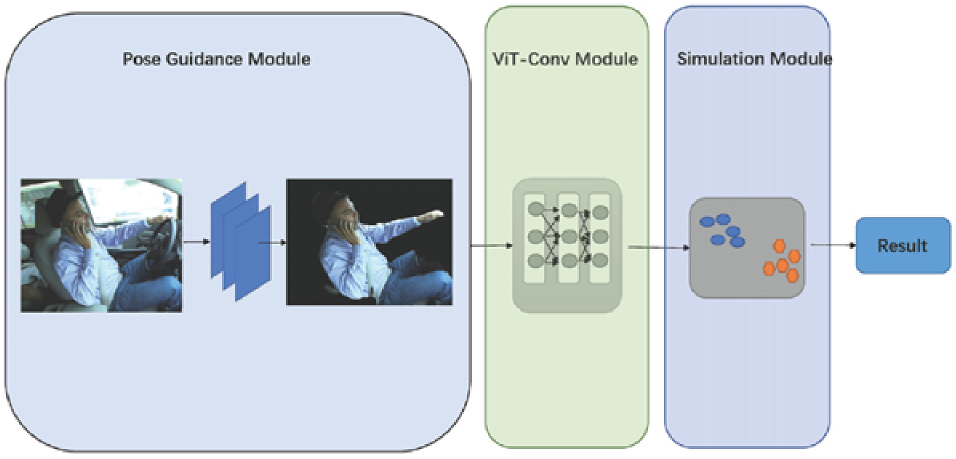


Figure 3.4: An overview of the proposed solution in [28]

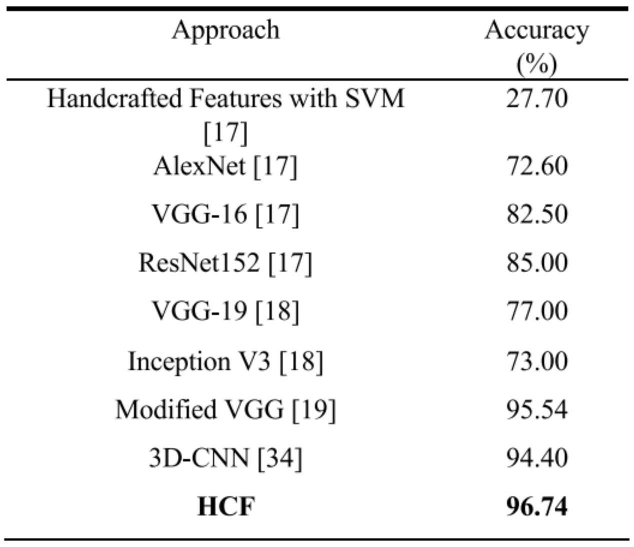
The following describes how the network functions as a whole:

- In the pose guidance module, from the segmentation of image we get a sub-image along with the driver and the elements that fall within his range of interactions.
- After that, this sub-image is an input to the Vit-Conv Module which has Conv and transformer blocks. Through their combination global and local features and image's feature map are extracted.
- Finally, the extracted feature map is an input into a simulation module and then the image classifications result at the inference stage is recorded.

3.2 Detection Using CNN Based Architectures

These papers basically aim to accurately identify behaviors of distracted driving and provide a mechanism for assessing the secondary tasks (like chatting on a cell phone) using a Hybrid Convolutional Neural Network Framework (HCF). It also proposes solutions to detect the distraction of drivers to avert possible accidents using different convolutional neural network (CNN) models.

They introduced a hHybrid CNN Framework (HCF) in this paper [22] to identify the behaviors of distracted driving. Three cooperating pre-trained CNN models extract features at various sizes. The parameters were received by pre-training ResNet50, InceptionV3, and Xception models on ImageNet. These parameters were immediately transferred to their detection job. The properties of the dataset of the State Farm differ from the dataset of ImageNet. Due to that reason it is mandatory to fine-tune pre-trained models so that it is able to fit the dataset of State Farm. The recommended HCF training technique is: For the pre-trained ResNet50, from the start to layer 151, weights were frozen and only those weights that range from layer 152 to the top layer were trained. After that, in InceptionV3, the weights were frozen for layer 0 to 171, and only the weights of layer after 171 to the top layer were trained. Finally, in Xception, pre-trained weights ranging from layer 0 to 116 were frozen and weights after that to the top layer were trained. They employed the (Multi Threshold-based Remora Optimization)MTRO method during training. The initial rate of learning and the batch size of image was set to respectively 0.001 and 64. Dropout was used to reduce overfitting. To finish the feature classification, a softmax classifier is employed. The final output is the probability which corresponds to the ten different distracted driving behaviors in the State Farm dataset. The features are then combined to produce the feature maps. The fully linked layer is then trained to define each type of distracted behavior. They use dropout technology during the training phase. It prevents the training model from becoming overfit to the training data. The results of distracted detection regions are highlighted with the use of (Class Activation Map)CAM. The results reveal that the proposed HCF performs well in identifying the distracted driving behaviors and has an accuracy of 96.74% in classification.



Approach	Accuracy (%)
Handcrafted Features with SVM [17]	27.70
AlexNet [17]	72.60
VGG-16 [17]	82.50
ResNet152 [17]	85.00
VGG-19 [18]	77.00
Inception V3 [18]	73.00
Modified VGG [19]	95.54
3D-CNN [34]	94.40
HCF	**96.74**

Figure 3.5: Result Analysis in [22]

In this paper[11], , works with a simple. SGD optimizer is built using 0.01 learning rate and momentum of 0.9. The photographs are modified to avoid overfitting and the model is run for a total of 20 epochs. The model gives the following output: After 20 epochs, the accuracy of training reaches 91.20% and validation accuracy is 54.45%.

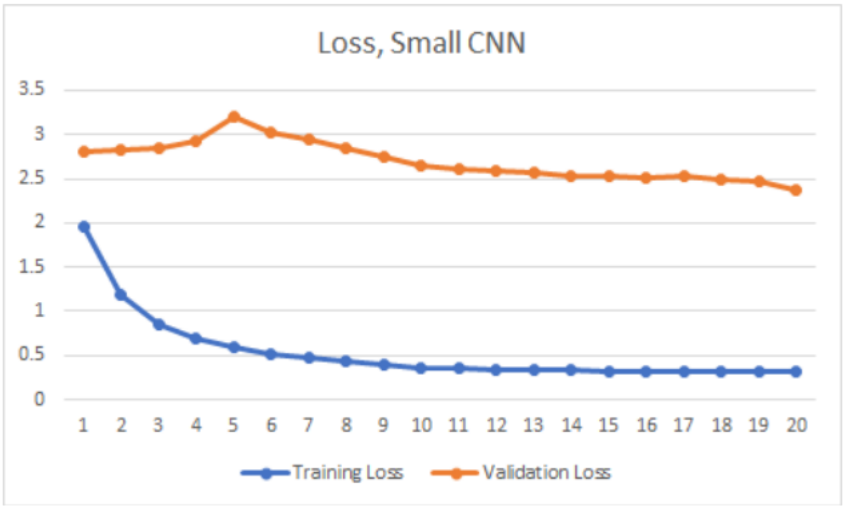
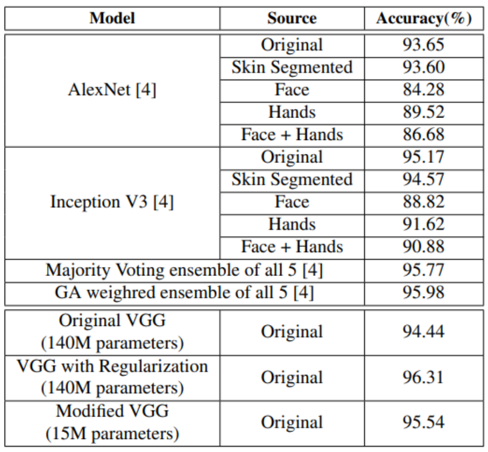


Figure 3.6: Overview of Training Loss in small CNN in [11]

The results shown here [11] denotes that the compared to validation accuracy training accuracy is much higher which implies overfitting.

In this paper, [11] Based on these findings, it became clear that the training accuracy is quite significantly greater than the validation accuracy. Even though it was an overfitting model, a dropout layer was used to solve the overfitting problem. They have to employ Google Cloud Platform for training purposes in this article. As not enough GPUs were available, they chose a batch size of 32. The model was unable to perform well with the initial learning rate which was 0.001. AS a result, they had to increase the learning rate up to 0.0001. They chose to train VGG-16 but only the top block. Then, they removed VGG-16's last Softmax layer and the Global Average Pooling Layer replaced it. Then, to overcome overfitting, as always, a dropout layer was added. Then, the final prediction values were generated by a fully connected layer. The results show that this model's validation accuracy reached around 80 to 83% after being trained for 2 to 4 epochs. Because the train- ing data was so little, the model overfitted. They chose to develop an ensemble of VGG-16 models to overcome overfitting. Thus, a second VGG-16 model of 64 batch size was used for the training. This time the validation accuracy was approximately 75 to 77%. VGG-19 was treated in the same way. Now for INceptionV3 which consists of 313 layers, an ImageNet database was used. In the distracted driving situation, only the top two blocks are trained. Adams Optimizer was used to build this model. However, when SGD optimizer was used instead if Aadm Optimizer, with the same learning rate, gave superior results. Batch normalization is known as a pre-processing procedure. Because in this case the weights of the top layers were initialized arbitrarily, so only they were trained in the beginning. Other models are non-trainable in this step. The validation accuracy achieved here is 0.73. As overfitting is a major worry in this topic, it was discovered in this paper that employing a combination of several models will generate improved results than using only a particular model. So, the final prediction values were obtained by calculating the average of the results of the combined models. The log loss value for the best VGG-16 model was 0.8157. Similarly, VGG-19 had a log loss value of 0.9632 and InceptionV3 had 1.0972. Combined log loss value was 0.795.

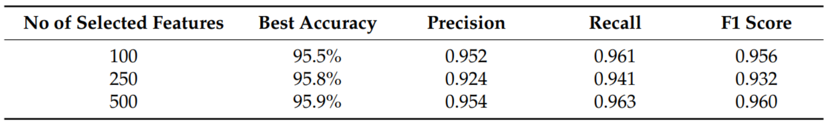
In one paper[10], a model of ImageNet which is pre-trained is used to initialize weight. The weights of each and all layers are modified in relation to the dataset. All of the hyperparameters have been fine-tuned after extensive testing. For training Stochastic Gradient Descent with the use of learning rate, a decay rate and a momentum value. The batch size was 64 and the epoch count was 100. Training and testing required high specifications. Thus, the graphics card was NVIDIA P5000 memory was 16 GB RAM. Frameworks have been created by Theano and Keras.In the training set, VGG-16 gives full accuracy and in tests it gives approximately 95%. Batch normalization can take this accuracy to around 97%. To process enough photos every second, the VGG-16 needs high memory and with a nearly 90 percent reduction in parameters while maintaining accuracy. On the test set, they scored an accuracy of 96.31%.



Model	Source	Accuracy(%)
AlexNet [4]	Original	93.65
	Skin Segmented	93.60
	Face	84.28
	Hands	89.52
	Face + Hands	86.68
Inception V3 [4]	Original	95.17
	Skin Segmented	94.57
	Face	88.82
	Hands	91.62
	Face + Hands	90.88
Majority Voting ensemble of all 5 [4]		95.77
GA weighred ensemble of all 5 [4]		95.98
Original VGG (140M parameters)	Original	94.44
VGG with Regularization (140M parameters)	Original	96.31
Modified VGG (15M parameters)	Original	95.54

Figure 3.7: Overview of Result Analysis in [10]

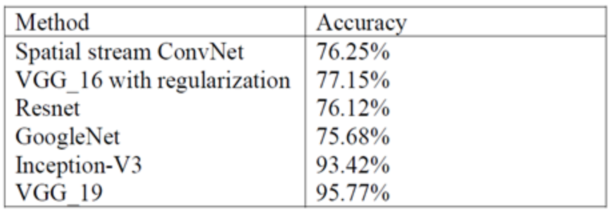
In this paper[22], while creating larger networks, EfficientNet outperforms earlier object detection models in terms of speed and success rate the parameters get increased all at once. EfficientDet's progress is extremely valuable to object detection research and application development. The average accuracy of this model is nearly 100% (99.16). When compared to other models, this model gives higher accuracy than any other models. So, according to the tests and researchers, this is the best distraction detection model. HOG is a feature descriptor that concentrates on the object's shape and structures. There are 2 sorts of feature fusions in their suggested method and the performance is measured through this. They employed KNN (K-Nearest Neighbor) and SVM (Support Vector Machine) variations of two different classifiers. There are in total 6 SVM versions. They are cubic, quadratic, linear, coarse, medium and fine Gaussian. On the other hand, KNN has 4 versions which are cosine, coarse, medium, fine. According to their analysis of their trials, performance and the number of features is proportionate until a certain point. After that the improvement pace also goes down. Efficient and accurate prediction graphs for 100, 250, and 500 features using variations of KNN and SVM classifiers. SVM-based classifiers have an accuracy range of 54.7% to 94.8% for 100 features. SVM-based classifiers have an accuracy range of 38.1 percent to 95.1 percent for 250 features. For 500 features, the range varies from 28.8% and 95.1%. For KNN, the accuracy range is 56.9% to 95.5% for 100 features and 250 features, the accuracy range is from 37.6% to 95.8%. So, every time the number of features increases the range of accuracy also increases. Except for the linear SVM, which has the fastest prediction rates for 100, 250, and 500 features, the prediction speeds of KNN versions are often faster than SVM variants.[25]



No of Selected Features	Best Accuracy	Precision	Recall	F1 Score
100	95.5%	0.952	0.961	0.956
250	95.8%	0.924	0.941	0.932
500	95.9%	0.954	0.963	0.960

Figure 3.8: Overview of Result Analysis in [25]

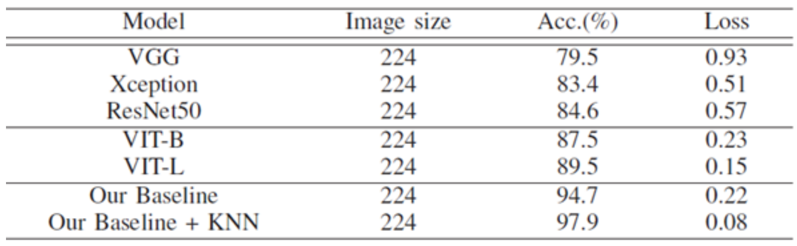
[htbp] This paper[19] analyzes the segmentation of parts of the human body and then describes the training and classification model and methods that were used for classification on the segmented image. VGG-19 and Inception-v3 are the two CNN models that have been explored in their study. Pre-trained model VGG-19 is made up of 19 layers and has quite small receptive fields (3×3).It was known for large-scale image recognition and was among the well-known models that were submitted to the ILSVRC-2014 challenge.Convolutions, average and maximum pooling, concatenations, dropouts, and fully connected layers of symmetric and asymmetric building are some of the building blocks that make up the other model known as InceptionV3. This model makes heavy utilization of batch normalization and applies these to activation inputs. The Softmax function is used to calculate the loss. The results after the re-implementation of the VGG-16 along with its regularization and Resnet and GoogleNet networks on their dataset are as following:



Method	Accuracy
Spatial stream ConvNet	76.25%
VGG_16 with regularization	77.15%
Resnet	76.12%
GoogleNet	75.68%
Inception-V3	93.42%
VGG_19	95.77%

Figure 3.9: Overview of Result Analysis in [19]

This paper[28] compares our suggested approach for assessment with a number of other currently existing methods. They only consider the recommendations and standards of ViTConv as the benchmark and while their finalized version of the model combines every other module. For the dataset of StateFarm, they evaluate different types of postures' images of the driver while driving. Their standard approach has an accuracy rate of 94.7%.SPACE Additionally, the whole model is 97.9% accurate, which performs far better than every existing state-of-the-art technique and boosts the base accuracy by 3.2%. Compared to other approacher, their method allows to notice the desired region of interest while using ViT-Conv in SFD3 dataset, which helps to detect different driving patterns.



Model	Image size	Acc.(%)	Loss
VGG	224	79.5	0.93
Xception	224	83.4	0.51
ResNet50	224	84.6	0.57
VIT-B	224	87.5	0.23
VIT-L	224	89.5	0.15
Our Baseline	224	94.7	0.22
Our Baseline + KNN	224	97.9	0.08

Figure 3.10: Overview of Result Analysis in [28]

3.3 Major Findings and Scope of Research

After a thorough review of previous publications that applied both machine and deep learning models to identify distracted drivers, there were four key discoveries that would serve as the foundation for this paper:

- The dataset[15] in this paper[22] only contains photos from the driver's right-handed part. Given the aforementioned constraints, we will examine driving style from other camera directions. In the future, we aim to reduce calculation time and parameter count not only to recognize driver's distraction but also to avoid distracted driving.
- In this paper[11] final output can be generated using KNN nearest match of the given sample considering the average of the probabilities [7]. Utilizing ResNet- 50 and -152 CNN models better results can be accomplished in this problem as these models are mostly known for their capability in terms of image classification problems. Better results can be achieved if unnecessary segments can be removed from sample images.
- We can decrease parameters and the time it takes to compute them [10]. So we would like to build a system that can identify both visual and manual distractions which is the major goal of our research.
- By incorporating new sensory modalities [22], may suggest enhancing the distraction detection system. For instance, we may employ a mike to capture the volume and noise in the car. That will detect different distracted driving ways. Also enhancing the models after including dynamic details.
- This domain[8] may be further investigated using cutting-edge techniques such as quantum computing. [25]
- We can design our unique CNN architecture to determine decent ROI. [19]

Chapter 4

The Dataset

Due to the advancements of Automobile industry, detection of distracted drivers has received massive attention from researchers and organizations who are particularly interested in classifying distracted driving and normal driving. Because of this numerous data have been collected by various researchers. However, not all datasets are publicly available. Kaggle organized a competition with the goal of collecting sufficient data for this task.

4.1 Data collection

HOWDRIVE 3D : Driver Distraction Dataset: With the purpose of studying distracted driving phenomenon, collected data by having drivers practice various driving behaviors within a car. This experiment was conducted through 9 drivers where everyone was instructed to carry out the 10 selected tasks independently. These 10 tasks are 10 classes of the dataset that help us to identify distracted driving and normal driving. For each class there are around 450 images of each driver and in total 9 drivers contributed to making of a single task. Every recording session conducted had different conditions based on the time of the day, clothes they wore, cars they used etc. The data was initially gathered in video format, then divided into 1080x1920 individual frames. An average of approximately 38 thousand images were collected after the manual assessment.

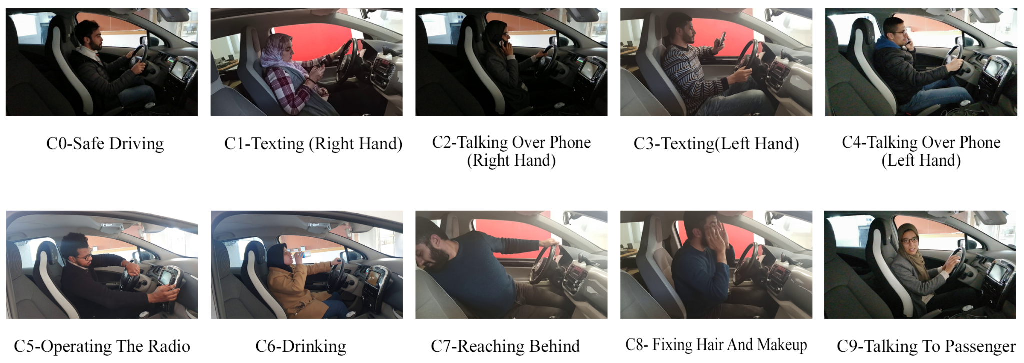
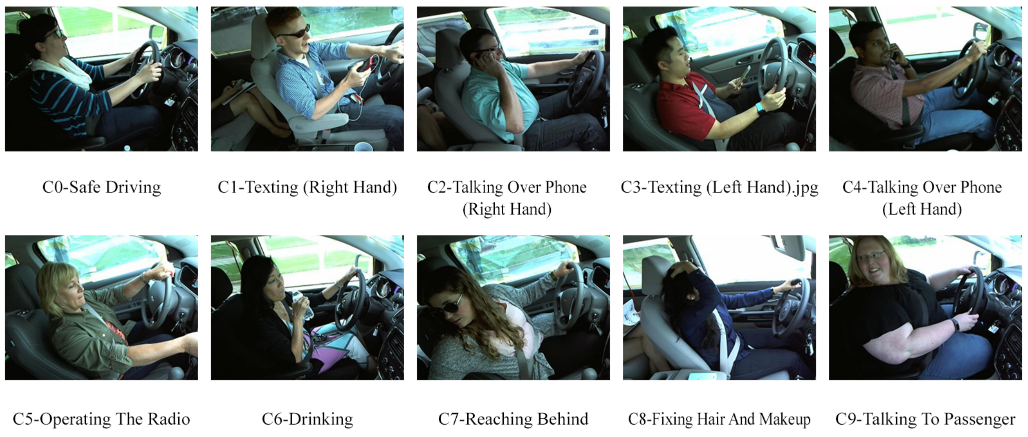


Figure 4.1: Different classes of Dataset Howdrive 3D : Driver Distraction Dataset

State Farm Distracted Driver Dataset (SFD3): This dataset won the competition organized by Kaggle which has 10 classes. and every class contains approximately 2000 images of dimension 320 x 240. In the field of distracted driving classification, this has been one of the most popular data that has been used in various researched. This dataset was utilized to test our proposed model apart from Howdrive Distracted Driver Dataset.



C0-Safe Driving
C1-Texting (Right Hand)
C2-Talking Over Phone (Right Hand)
C3-Texting (Left Hand).jpg
C4-Talking Over Phone (Left Hand)
C5-Operating The Radio
C6-Drinking
C7-Reaching Behind
C8-Fixing Hair And Makeup
C9-Talking To Passenger

Figure 4.2: Images for all classes of SFD3 Dataset

4.2 Data Analysis

For this work we chose Howdrive Distracted Driver Dataset that contains 38000 frames in total. The dataset divided into several categories (10 classes) of distracted driving behaviors named from c0 to c9. All these classes contained information about the different types of distraction and the number of times these distractions occurred during all these sessions. They were categorized into these- Safe driving, Texting with right hand, Talking on Phone-Right hand, Talking on Phone-Right hand, Texting with left hand, Adjusting radio, Drinking, Fixing hair/doing makeup, Reaching behind, Talking to passenger.

classes	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
Activity	Safe Driving	Texting-left hand	Texting-right hand	Talking on the phone-left hand	Talking on the phone-right hand	Operating the radio	Drinking	Reaching behind	Fixing hair and makeup	Talking to passenger
Number of Images	3879	3792	3959	3807	4018	3853	3629	3849	3791	3750

Table 4.1: Table of class and images

4.3 Data Pre-Processing for Models

Prior to using images in models, they must be pre-processed to guarantee that there are no biases or contradictions in the predictions made by the models due to the nature of the data.

4.3.1 Making Sub-Dataset

At the early stages of model implementation, the large dataset can be the reason for massive time consumption and heavy computational cost. That's why the original dataset has been divided into two sub-dataset versions where Dataset Version-1 contains half images per class compared to the original dataset and Dataset Version-2 contains around one-tenth of images compared to the original dataset.

Activity	Safe Driving	Texting-right hand	Talking on the phone-right hand	Texting-left hand	Talking on the phone-left hand	Operating the radio	Drinking	Fixing hair and makeup	Reaching behind	Talking to passenger
Classes	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
Original Dataset	3879	3792	3959	3807	4018	3853	3629	3849	3791	3750
Dataset V1	1035	1011	1056	1015	1072	1028	968	1027	1011	1000
Dataset V2	552	549	593	594	590	525	550	535	579	563

Table 4.2: Table of Sub-Datasets

4.3.2 Data Augmentation

As stated in the earlier section, the dataset has been made smaller for the ease of implementation process. However, when a neural network model is trained on a small dataset, the model often shows a tendency of overfitting the model by memorizing the data. The lesser samples are provided for training, the more models tend to do overfitting. Thus, before being fed into the model the dataset needs to be pre-processed with several data augmentation techniques.

1. **Resize to 224x224 pixels:** From the original pixelsimages were scaled down to 224x224 pixels so that the complexity of working with a large array of pixels can be minimized.
2. **Zoom:** Applied 0.2 zoom range.
3. **Brightness Range:** Brightness range has been kept (0.9,1.5) to slightly darken the image.
4. **Height shift:** Height shift was applied in images for a clear view.
5. **Width shift:** Height shift was applied in images for a clear view.
6. **Rotation:** The images were rotated anticlockwise by 10 degrees.

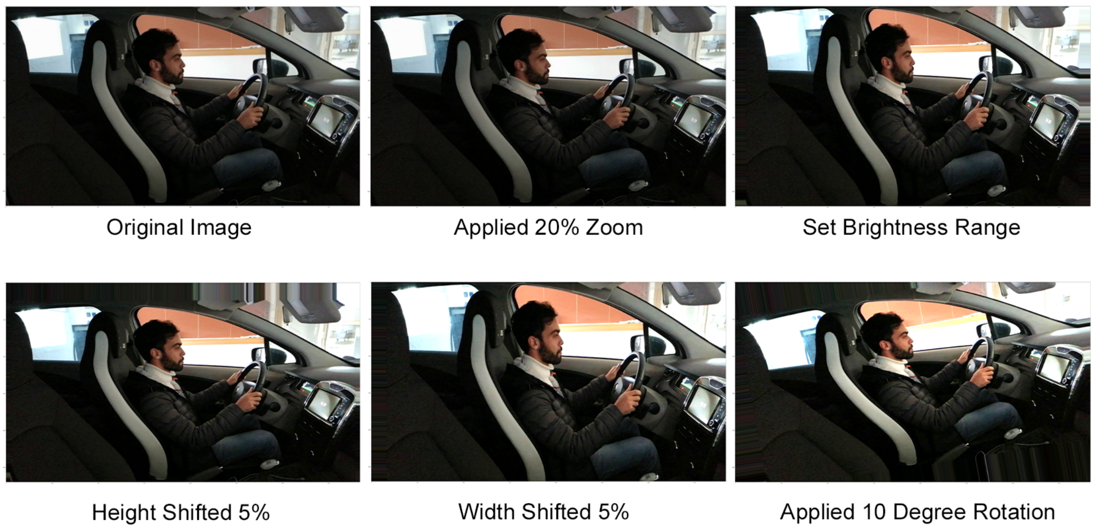


Figure 4.3: Images After Augmentation

Chapter 5

The Models

After performing necessary pre-processing on the dataset, it is ready to be implemented on a model. Later on, we divided the entire dataset into three subsets. Using SciKit-Learn's test-train-split() method, the dataset was split into 80:20 ratio for training, testing respectively. Then 10% of the training set has been used for validation which is also 8% of the total dataset. The training subset is used to train this portion of the dataset based on the model it has been fed into and the test subset is used to evaluate the result based on its learning after the chosen data has been trained by the model. The parameter "random-state" has been set to make sure that the data assigned to our train and test subsets is set randomly. [22]Also, the "Stratify" parameter has been specified to have a balanced number of examples for each class label. Figure-3.1 represents the segmentation of the dataset used for testing and training before implementing on the model.

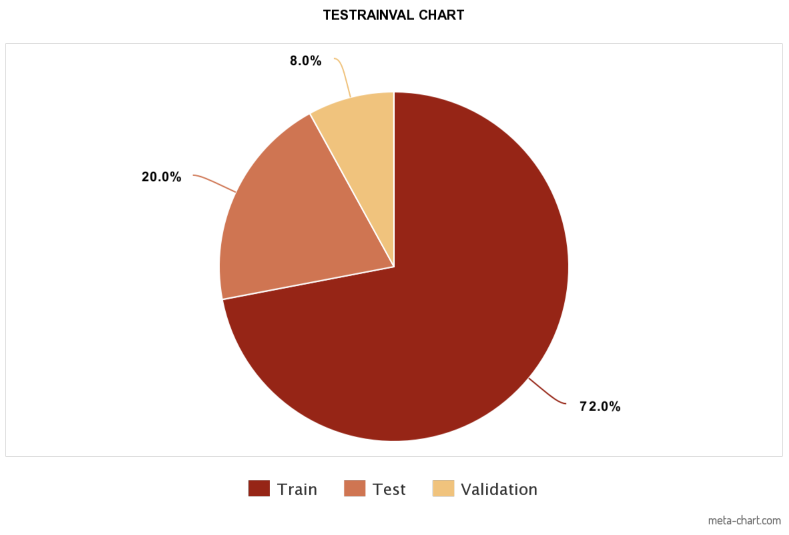


Figure 5.1: Dataset Segmentation for Training and Testing Models

5.1 Transfer Learning Model

Transfer learning enables use of pre-trained models which resolves the issue of the massive computational cost and time resources required to develop models for large datasets. The transfer learning models that have been used in this work were trained on the ImageNet dataset, an image database organized according to the WordNet hierarchy in which each node of the hierarchy is depicted by hundreds and thousands of images. Transfer learning is often used in deep learning due to the substantial resources needed to train deep learning models. Among the three kinds of methods of using transfer learning models, fixed feature extraction was used. In order to use the pre-trained model as a feature extractor, the fully connected layer is removed while the weights in the feature extraction layer remain frozen. After implementation of several models through the TensorFlow library, only best performing models were selected for further processing.

5.1.1 Resnet50

Resnet50 architecture contains 50 layers among which 34 are convolutional, 1 is maxpool and 1 average pool layer. There are 5 stages in the ResNet-50 architecture, each containing a convolution and an identity block and 3.8 x 10$\hat{9}$ Floating points operations. Each identity block and convolution block have three convolution layers. ResNet performs identity mapping via shortcut connections that bypass one or two layers to avoid the increased inaccuracy that results from simply adding layers to an existing network. This skip-connection technique helps to reduce the issue of vanishing gradient. There are around 23 million trainable parameters in the ResNet-50.

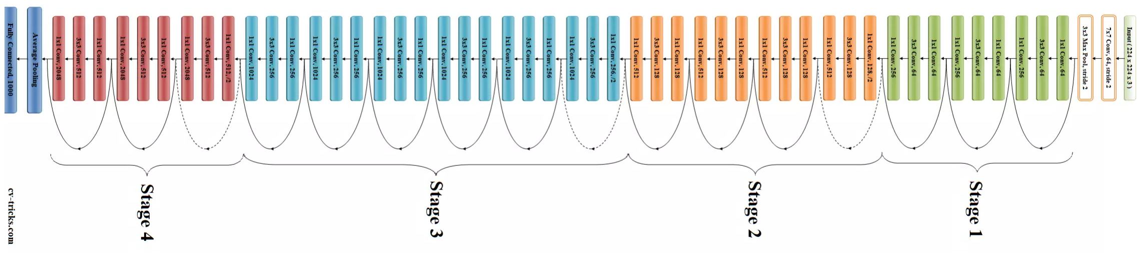


Figure 5.2: Architecture of ResNet50

1. A convolution with a kernel size of 7 * 7 and 64 different kernels all with a stride 2.

2. Max pooling with stride size of 2.

3. Convolution layers with 64 1x1 kernels, 64 3x3 kernels and 256 1x1 kernels. These three layers are repeated 3 times which sums up to 9 layers in this step.

4. Convolution layers with 128 1x1 kernels, 128 3x3 kernels and 512 1x1 kernels. These are then repeated 4 times.

5. Convolution layers with 256 1x1 kernels. This step repeats 4 times, giving us 12 layers in this step.

6. Convolution layers with 256 1x1 kernels, 256 3x3 kernels and 1024 1x1 kernels, which too are repeated for 6 times creating a total of 18 layers.

7. Convolution layers with 512 1x1 kernels, 512 3x3 kernels and 2048 1x1 kernels, repeated 3 times. giving a total of 9 layers.

8. An average pooling layer connected to a fully connected layer containing 1000 nodes having a softmax function at the end. Which again makes 1 layer.

For this research, while connecting to the final output layer with 1 node and softmax activation function, 256 hidden nodes with ReLu hidden nodes and dropout of 0.5 were set. Now the model has Total parameters of 24,144,826 Trainable parameters are 527,114. The model was compiled using Adam optimiser while setting learning rate as 0.001, as loss function categorial-crossentropy was set and accuracy as metric. Various parameters were used while training and testing the model. However, the best accuracy was achieved after training the model for 10 epochs, while keeping 155 steps per epoch.

5.1.2 ResNet50V2

ResNet50V2 is a modified version of ResNet50 that is usually seen performing better than ResNet50 on the ImageNet dataset. In ResNet50V the propagation formulation of the connections between blocks has been altered. This version of ResNet50 uses the pre-activation of weight layers instead of post-activation removing the path of the input to output in the form of identity connection and applied Batch Normalization and ReLU activation as input before performing convolution operation.

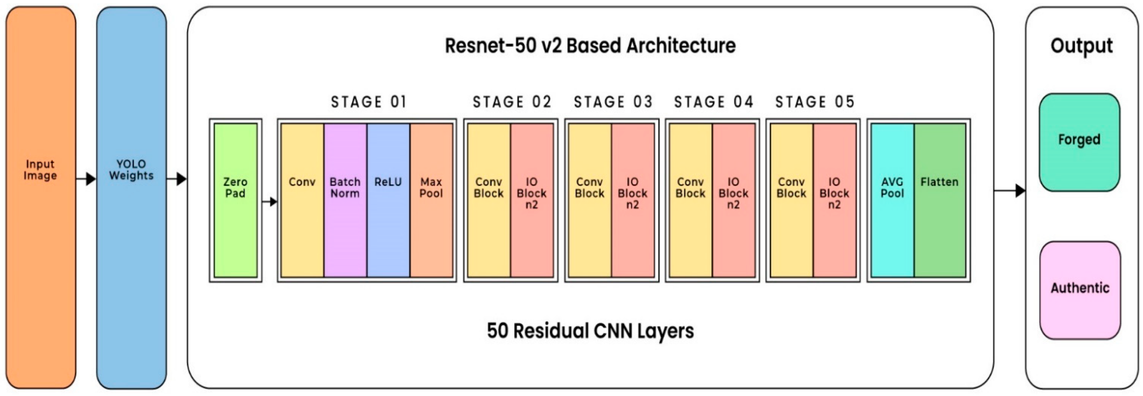


Figure 5.3: Architecture of ResNet50

For this research, the model was compiled using Adam optimiser. Categorial-crossentropy was set as loss function and again, accuracy as metric. After using several parameters for training and testing the model, the best accuracy was achieved after training the model for 5 epochs, while keeping 254 steps per epoch.

5.1.3 VGG16

VGG16 is a type of CNN (Convolutional Neural Network) often used for object detection and classification algorithms. It is a widely used technique for classifying images and is simple to employ with transfer learning. This model is able to significantly reduce the number of weight parameters. This may also be viewed as a regul[16]arization of the 7 x 7 convolutional filters that makes those fit through the 3 x 3 filters, with added non-linearity in between by using ReLU activations. By doing this, the network's propensity to over-fit during the training diminishes massively.

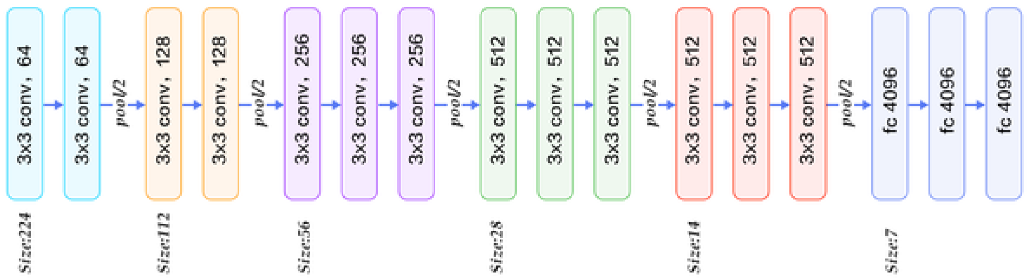


Figure 5.4: Architecture of VGG16

1. A convolution layer with 64 3x3 kernels, with a stride of 1, and repeated 2 times.
2. A convolution layer with 128 3x3 kernels, with a stride of 1, and repeated 2 times.
3. A Convolution layer using 128 filters 3x3 kernels, repeated 2 times
4. A convolution layer with 256 3x3 kernels, with a stride of 1, and repeated 3 times.
5. convolution layer with 512 3x3 kernels, with a stride of 1, and repeated 3 times.
6. A convolution layer with 512 3x3 kernels, with a stride of 1, and repeated 3 times.
7. A MaxPool layer of 3x3 kernel size and stride of 2 after each group of convolution layers.
8. 2 fully connected layers with 4096 hidden nodes.
9. fully connected output layer with 1000 hidden nodes, followed by a softmax activation layer.

Before connecting to the output layer with sigmoid activation function, flattening the previous outputs to a one-dimensional vector has been done. The last fully connected layer has been replaced by a fully connected layer with 64 hidden nodes using ReLU activation and dropout 0.5. This allowed to build a model with almost 11,965,578 trainable parameters and around 17 million non-trainable ones [17]. After making changes in several parameters the maximum average results after training the entire model for 5 epochs with the default steps per epoch of 155, repeating it for 4 times.

5.1.4 Inception-ResNetV2

Inception-ResNet-v2 is a convolutional neural based model that is trained on images from the ImageNet database. In the Inception-ResNet model, batch-normalization is used only on top of the traditional layers. Each Inception block has a filter 1×1 convolution without activation that scales up the dimensionality of the filter before matching the depth of the input. Inception-ResNet-v2 has 164 layers. The dimensions of the input from the previous layer and the output from the inception module must match for this to function. Therefore, for Inception-ResnetV2 factorization is crucial to match dimensions. The network by default receives a 299*299 pixel picture as input, and it generates outputs of a list of estimated class probabilities. It is constructed using both the Residual connection and the Inception structure. In addition to avoiding the degradation issue because of the deep structures, using residual connections shortens training time.

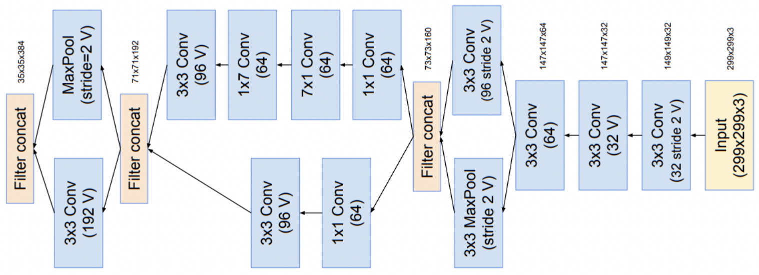


Figure 5.5: Architecture of Inception-ResnetV2

Similar to previous models, this model was also compiled using Adam optimiser keeping the learning rate as 0.001. We have set categorial-cross-entropy as loss function and accuracy as metric. After many trials and errors, the best accuracy was obtained after training the model for 20 epochs, while keeping 190 steps per epoch.

5.1.5 InceptionV3

The layer depth of the convolutional layer InceptionV3 is 48. This model achieved more than 78.1% accuracy on the dataset of ImageNet in terms of image recognition. Convolutions, average and maximum pooling, concatenations, dropouts, and fully connected layers of symmetric and asymmetric building are some of the building blocks that make up InceptionV3. The model makes substantial use of batch normalization, which is applied to activation inputs and computes the loss using Softmax. The architecture of this model is the same as InceptionV1.

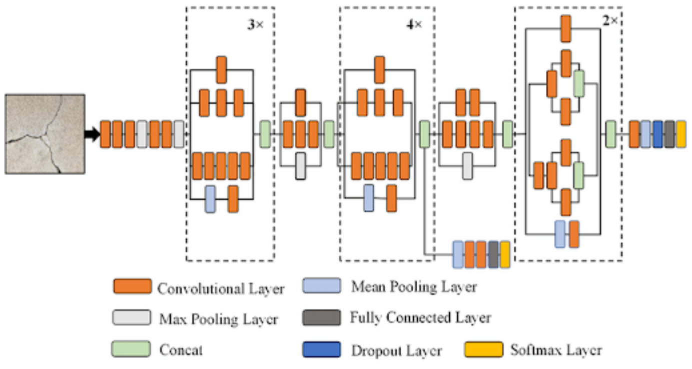


Figure 5.6: Architecture of InceptionV3

The inception V3 is just the advanced and optimized version of the inception V1 model. The Inception V3 model used several techniques for optimizing the network for better model adaptation. They are:

1. Factorized Convolutions: It monitors the network efficiency and lessens the number of parameters used in a network and as a result, the computational efficiency gets reduced.

2. Smaller Convolutions: It takes less time to train when the smaller convolutions are used in place of bigger convolutions. For example, if two 3x3 filters are used instead of a 5x5 filter the 3x3 filters will have 3x3 + 3x3 = 18 parameters whereas the 5x5 filter would've had 5x5 = 25 parameters.

3. Asymmetric Convolutions: In this case, let, a 3x3 convolution has been substituted by a 1 x 3 convolution and 3 x 1 convolution. But if it's replaced by a 2x2 convolution, the number of parameters becomes significantly more than in the proposed asymmetric convolution.

5.1.6 Xception

The Xception model is Extreme Inception which is an inspired form of the Inception model of CNN. It consists of deep and extensive convolutional layers working parallelly. The Xception model is 71-layer deep.

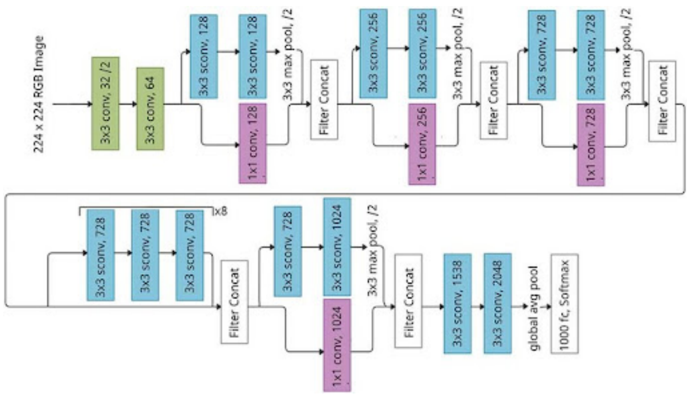


Figure 5.7: Architecture of Xception

The architectures are described below:

The model has two different levels with three convolutional layers on both of these levels. One of these levels has a single layer.

The output is divided into three pieces by this layer before moving on to the next set of filters.

The filter of the first level is a single convolutional level of 1x1. The next level uses 3x3 filters and three convolutional levels.

The architecture of this model is loaded with convolutional layers that are separable based on the depth. The training of the pre-trained version of this model is done using millions of images from the database of ImageNet. In addition, this model offers rich utility representations for a variety of images and can categorize hundreds of different item categories. This model excels in categorization and identification of images.

5.1.7 DenseNet201

As the name suggests, DeseNet201 convolutional network is 201 layers deep. From the ImageNet database, the networks' pre-trained version which has been trained on more than a million images can be loaded and this model will be able to classify objects like animals, pens etc 1000 other objects. This gives the network rich feature categorization for a lot of different types of images. The input size of the image is 224 by 224.

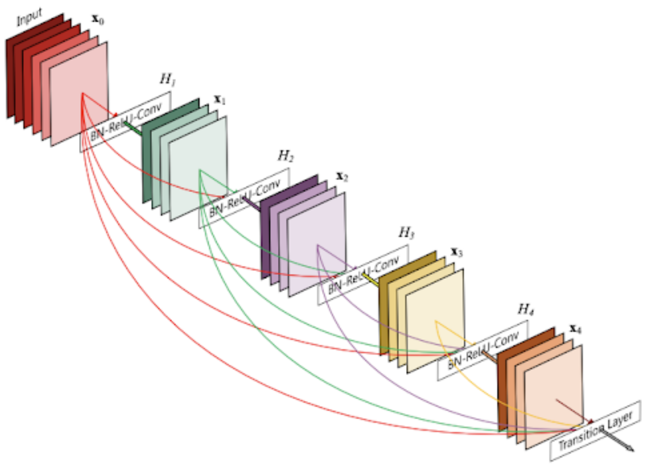


Figure 5.8: Architecture of DenseNet201

The DeseNet201 architecture is given below.

For each composition layer, Pre-Activation Batch Norm (BN) and ReLU, then 3×3 Conv are done with output feature maps of k channels.

To reduce the model complexity and size, BN-ReLU-1×1 Conv is done before BN-ReLU-3×3 Conv.

1×1 Conv followed by 2×2 average pooling are used as the transition layers between two contiguous dense blocks.

Feature map sizes are the same within the dense block so that they can be concatenated together easily.

At the end of the last dense block, a global average pooling is performed and then a softmax classifier is attached.

5.1.8 MobileNet-V2

MobileNetV2 consists of 53 convolutional layers that makes it a very effective and efficient feature extractor for object detection and segmentation. It has higher classification accuracy with fewer parameters.

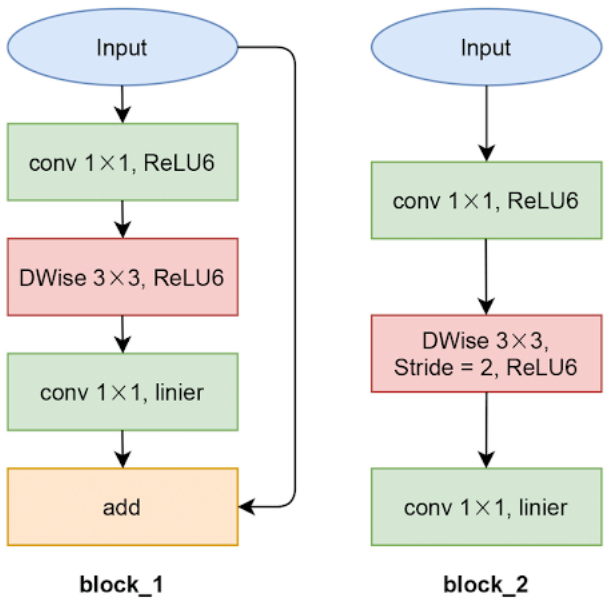


Figure 5.9: Architecture of MobileNetV2

Architecture:

1. It has 2 types of blocks- a residual block with 1 stride and a block for downsizing with stride 2.

2. Three layers for each block.

3. First layer is 1×1 convolution with ReLU6.

4. Convolution in the second layer is based on the depth.

5. The third layer is another 1x1 layer like the first layer but does not have any non-linearity. According to some arguments, the deep networks only have a linear classifier on the output domain's non-zero volume part when ReLU is used again.

6. Expansion factor is defined as "t" and t=6 for all the primary experiments. For example, if the number of input channel is 64, the internal output will have 64 x t = 64 x 6 = 384 channels

5.2 Convolutional Neural Network Models

Building a unique CNN was one of the key goals of this study. While training the Transfer learning model we have noticed that we achieve highest accuracy if the model is trained with fewer parameters. As we have discussed previously, most of the transfer learning models have a large amount of parameters. The goal was to create inexpensive, simplistic models that required fewer training parameters but had improved accuracy than the pre-trained models that we've employed in this study. We constructed two unique CNN models in order to achieve this objective.

5.2.1 3-Layer CNN Model Architecture

The custom 3 layer CNN model that was built and trained has an architecture as follows :
1. A convolution layer that includes 64 3x3 sized kernels, with a default stride , ReLU activation, outputting shape of 224x224x64 and 1 zero padding. The output parameter for this conv2D layer is ((3*3*3)+1)*64 = 1792

2. A Maxpool2D layer with 2x2 size that has a default stride of 2 with an output shape of 112x112x64.

3. A Dropout of 10% has been implemented to avoid overfitting.

4. A convolution layer with 64 3x3 sized kernels including default stride while ReLU is the activation function, giving us an output shape of 110x110x64. And the output parameter for this conv2D layer is ((3*3*64)+1)*64 = 36928.

5. A 2x2 sized maxpooling layer, with default stride of 2 was applied giving us an output shape of 55x55x64.

6. A dropout of 10% has been implemented again.

7. A convolution layer with 64 3x3 kernels, with a stride of 2, ReLU activation and an output shape of 53x53x64. And the output parameter for this conv2D layer is ((3*3*64)+1)*64 = 36928

8. A 2x2 size Maxpool2D layer with a default stride has been implemented which gives us an output shape of 26x26x64.

9. A dropout of 10% has been implemented.

10. A flatten layer has been added to convert the 2D outputting matrix into a linear vector.
9. A dense layer with 10 nodes along with softmax activation was implied, outputting only 1 value for class prediction. This is the output layer of our model.

Optimizer	Adam
Loss Function	Categorical Cross Entropy
Activation Function	Softmax
Batch Size	32
Dropout	10%
Total Parameters	508298

Table 5.1: Hyperparameters Of 3- layer CNN Model

5.2.2 6-Layer CNN Model Architecture

The proposed model has been constructed employing the Adam optimizer for all versions of the dataset while categorical-cross entropy has been specified for the loss function as we are working on a multi-class classification among 10 classes. The activation function that has been used is Softmax. Moreover, a dropout rate of 10% was set to prevent overfitting while setting the batch size was kept as 32.

Optimizer	Adam
Loss Function	Categorical Cross Entropy
Activation Function	Softmax
Batch Size	32
Dropout	10%
Total Parameters	1,793,514

Table 5.2: Hyperparameters Of 6 layer CNN Model

Block 1:

1. A convolution layer that includes 64 3x3 sized kernels, with a default stride of 2, “same” padding and ReLU activation, outputting shape of 224x224x64.The output parameter for this conv2D layer is ((3*3*3)+1)*64 = 1792.

2. A Maxpool2D layer with 2 x 2 pool size and stride size of 2 with outputting an image of 112, 112, 64 shape.

3. A dropout of 10% has been implied to reduce overfitting.

Block 2:

4. Another Conv2D layer that includes 64 3x3 sized kernels, with a stride of 2, “same” padding and ReLU activation, outputting shape of 110, 110, 64. The output parameter for this conv2D layer is ((3*3*64)+1)*64 = 36928

5. A Maxpool2D layer with 2 x 2 size and default stride size of 2 outputting an image of (55, 55, 64) shape.

3. A dropout of 10% has been implied to reduce overfitting

Block 3:

6. A Cnov2D layer layer with 64 3x3 sized kernels with stride of 2, “same” padding and ReLU activation function. It gives us an output of, ((3*3*64)+1)*64 = 36928 parameters. The output shape of this layer is (53, 53, 64)

7. A Maxpool2D layer with 2 x 2 size and stride size of 2 outputting an image of (26, 26, 64 shape).

8. A dropout of 10% have been added.

Block 4:

9. A Cnov2D layer layer with 128 3x3 sized kernels with stride of 2, “same” padding and ReLU activation function. It gives us an output of, ((3*3*64)+1)*128 = 73856 parameters. The output shape is, (24, 24, 128)

10. A Maxpool2D layer with 2 x 2 pool size and default stride size outputting an image of (12, 12, 128) shape to be passed on next layer.

11. A dropout of 10% have been set.

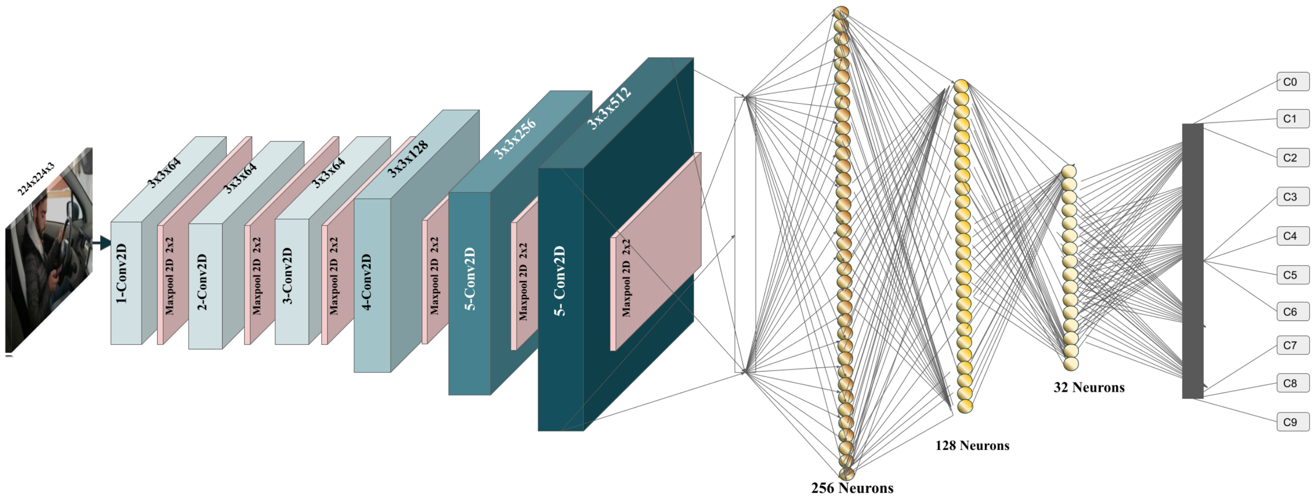


Figure 5.10: Architecture of 6 Layer Custom CNN

Block 5:

12. A Cnov2D layer layer with 256 3x3 sized kernels with stride of 2, “same” padding and ReLU activation function. It gives us an output of, ((3*3*128)+1)*256 = 73856 parameters and an output shape of 10, 10, 256

13. A Maxpool2D layer with 2 x 2 pooling size and stride size of 2 outputting an image of 5, 5, 256 shape. giving us an output shape of 24 x 24 x 128

14. A dropout of 10% have been set.

Block 6:

15. A Cnov2D layer layer with 128 3x3 sized kernels with stride of 2, “same” padding and ReLU activation function. It gives us an output of, ((3*3*64)+1)*128 = 73856 parameters. The output shape is 3, 3, 512

16. A Maxpool2D layer with 2 x 2 pool size and default stride size of 1 outputting an image of 1, 1, 512 shape. giving us an output shape of 24 x 24 x 128

17. A dropout of 10% have been set.

18. 3 fully connected layers have been added with 256, 128 and 32 nodes that gives us an output of 131328, 32896 and 4128 parameters respectively.

19. A flatten layer has been added to convert the 2D outputting matrix into a linear vector.

20. A fully connected output layer of 10 nodes have been added as we have 10 classes with Softmax activation function.

5.3 Model Evaluation

After years of research various metrics were found which are necessary to evaluate a model. These metrics help to understand the excellence of a model.

5.3.1 Confusion Matrix

Confusion Matrix is a method for analyzing the effectiveness of a classification algorithm or Supervised model. In a binary classification, may be 4 outcomes as below:

1. True Positive: Both predicted and actual outcomes are positive.
2. True Negative: Both predicted and actual outcomes are negative.
3. False Positive: Predicted outcome is positive but actual outcome is negative.
4. False Negative: Predicted outcome is negative but actual outcome is positive.

5.3.2 Accuracy and Loss

Accuracy:

Accuracy is an indicator of the model’s performance across all classes which quantifies how frequently the classifier predicts correctly. In other words, it is the percentage of estimates in which the predicted value and the actual value are the same. The accuracy metric is often inappropriate for datasets with unbalanced classes as the accuracy score would often be high. For classification issues that are properly balanced and not distorted, accuracy is a reasonable viewpoint to evaluate the model.

$$\text{Accuracy}=\frac{True_{positive}+False_{negtive}}{True_{positive}+True_{negative}+False_{positive}+False_{negtive}} \tag{5.1}$$

Loss:

A model's loss or cost function evaluates how well or poorly it performs after each optimization cycle. It is the summation of errors for each sample in the training and validation set. For diverse usage depending on the dataset, Keras and Tensorflow include a variety of built-in loss functions.

5.3.3 Training Vs Validation Accuracy

Training accuracy is obtained after the model has been run through the training sets and similarly validation accuracy refers to accuracy that's obtained after validation set has run accuracy. Training accuracy demonstrates a model's capability while training while validation accuracy shows the ability to adapt to new dataset. When both the accuracy are close to equal, the model shows no overfitting, but as the training accuracy gets higher than validation accuracy it shows signs of overfitting to training datasets, which means it is fitting to unnecessary noise while training. Similar to how training and validation losses may assist identify if a model is overfit, underfit, or neither can also be used to assess its quality. As a result, the model is roughly accurate when training loss and validation loss are comparable.

Precision, Recall and F1 Score

The precision is the ratio of the total predicted positive instances to the actual positive cases. The target of precision is to measure the ratio of the positively predicted labels being precise and that's why it's also called positive predictive value. To balance false positives and false negatives, precision is utilized in combination with recall. Precision depends on the class distribution. It's a measure of accuracy. High recall and precision models can be used to minimize false negatives. When the classes are highly imbalanced, the precision score is a defining aspect of how well predictions worked. The mathematical representation of this will be,

$$Precision = \frac{True_{positive}}{True_{positive}+False_{positive}} \tag{5.2}$$

On the other hand, recall is denoted for sensitivity and conveys the same meaning. Model recall signifies the model's ability to correctly predict the positives out of actual positives. It measures the credibility of the machine learning that has been used in terms of classifying all actual positives out of all positives that exist within a dataset. Higher recall score indicates model high ability at detecting positive examples and vice versa. To provide a comprehensive picture of the model's performance, recall is often used in association with other performance metrics like precision and accuracy. The mathematical representation of this will be,

$$Recall = \frac{True_{positive}}{True_{positive}+False_{negative}} \tag{5.3}$$

F1-score gives the same amount of weight to Precision and Recall for measuring its accuracy performance and makes it an alternative to Accuracy metrics. It's also used as a single value to get comprehensive details about the output quality of the model. It is the harmonic mean of accuracy and recall is crucial for unequal class distributions. This measurement is often useful when either precision or recall score need to be optimized in a decreased model performance. The mathematical representation is,

$$F1Score = \frac{2.Precision.Recall}{(Precision+Recall)} \tag{5.4}$$

Chapter 6

Result Analysis

This chapter demonstrates the results and findings found during this study while analyzing the results. After training the model results have been evaluated observing the metrics discussed in the previous section, the models' test accuracy, loss, and required parameters were assessed to provide a rough indication of how well the models have performed. Test accuracy and Test loss will provide an overview of how the models have executed on unseen segments of the dataset based on their learning on the training set. All of the models have been trained using training and validation portions of the dataset and evaluated using the test dataset portion. As we have mentioned previously, there are 3 versions of the dataset. Those are, Original dataset, Dataset- V1 and Dataset-V2. As previously stated in Chapter 5, we have used 7 transfer learning models and 2 custom CNN models in this work. All of those 9 models have been used to train and test 3 versions of the dataset. Original dataset has been trained on 30 epochs, Dataset-V1 on 70 epochs and Dataset-V2 on 100 epochs.
All of the models will be judged based on some factors. Those are, testing accuracy and loss, precision, recall, F1 score and parameters of the model. Another important factor that we should take into consideration is the parameters required for the model.

6.1 Analyzing Confusion Matrix, Accuracy & Loss

6.1.1 Dataset Version - 2

The smallest version of our dataset is indicated as Dataset-V2 that has been created for ease of implementation initially. Due to being the smallest dataset, it was easiest to train as it took less computational time to execute each epoch. However, the challenge of working with small datasets is, it is easy to be overfitted as in most cases models fail to learn the proper patterns of the inputs. Along with decreasing samples of the dataset there are higher chances of the wrong validation. Because of this it was necessary to use higher numbers of epochs as The results obtained from dataset-2 after running several pre-trained CNN models and customized CNN models are shown in the Table 6.1 below.

Pre-trained Models: We have applied 7 Transfer Learning models that are based on CNN-architecture. Test accuracy depicts the models' ability to differentiate be-

		Test Accuracy	Test Loss	Val(max) Accuracy	VAL(min) Loss	Avg Precision	Avg Recall	Avg F1 Score
Pre-Trained CNN Models	ResNet101	0.9858	0.0967	0.9816	0.1054	0.99	0.99	0.99
	Xception	0.9939	0.0522	0.9945	0.0489	0.99	0.99	0.99
	InceptionV3	0.9909	0.1239	0.9963	0.049	0.99	0.99	0.99
	InceptionResNetV2	0.9949	0.0182	0.9982	0.0106	1	0.99	0.99
	DenseNet201	0.9959	0.0203	0.9945	0.0584	1	1	1
	VGG16	0.9983	0.0139	1	0.0039	1	1	1
	MobileNetV2	0.9979	0.0205	0.9963	0.0563	1	1	1
Custom CNN Models	CNN - 3 layer	0.9961	0.0254	0.9875	0.0474	1	1	1
	CNN - 6 layer	0.9959	0.0122	1	0.0024	1	1	1

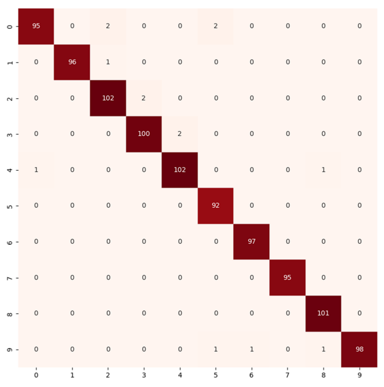
Table 6.1: Accuracy of Models in Dataset-V2

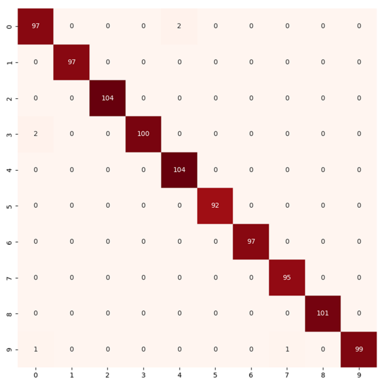
tween different distracted driving behavior from unseen data. If the test accuracy is high, then it means that the model can distinguish between the behaviors much more accurately. From the table -X we can see that, among all of the pre-trained CNN models, Resnet101 shows the worst performance while evaluating the model. "Test accuracy" indicates the results obtained after evaluation on the test set which is the unseen portion of the entire dataset. Resnet101 achieved test accuracy of 0.9858, which is the lowest among all Transfer Learning models. On the other hand, the highest accuracy was achieved by VGG16 which is 0.9983. Meanwhile, other pre-trained CNN models that showed better performance that Resnet101 are, Xception, InceptionV3, InceptionResnet-V2, DenseNet201 and Mobilenet, scoring accuracy of 0.9939, 0.9909, 0.9949, 0.9959 and 0.9979 respectively on test portion of dataset. The accuracy of these models varies slightly. DenseNet201 performs 0.1% better than InceptionResnet-V2 and Mobilenet's performance varies with Densenet201's by 0.2%.

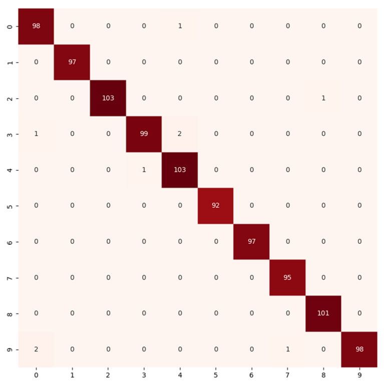
Test loss reflects the difference between the actual and predicted outcomes, and a low loss rate denotes a classifier's ability to classify with greater confidence. Among all pre-trained CNN models InceptionResNet-V2 performs the worst, scoring the highest test loss of approximately .1239. Therefore, ResNet101 shows a better outcome than Inception-Resnet-V2 obtaining a test loss of 0.0967. Meanwhile, Xception, Mobilenet, DenseNet201, InceptionResnet-V2 reached test loss of 0.0522, 0.0205, 0.0203 and 0.0182 respectively and VGG16 performed best achieving the lowest test loss.

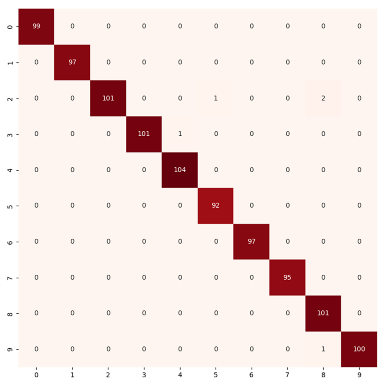
Among transfer learning models DenseNet201, VGG16 and MobileNetV2 achieve the highest precision, recall and F1 score. As described in Chapter-5, the confusion matrix for multi-class classification plots a NxN matrix where N is the number of classes. For this work, N=10, the amount of classes that we are classifying as we can see in figure 6.1.

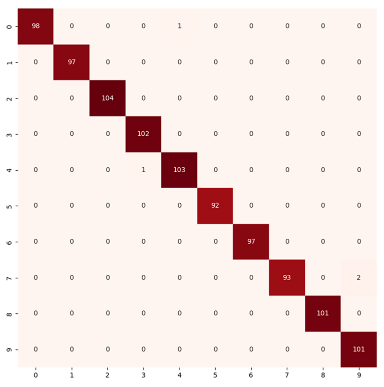
Custom CNN Models: Among the two custom CNN models that have been implemented, the first one is a 3 layer CNN model which consists of 3 convolution layers. As it can be observed from the Table 6.1 above, on the test dataset an accuracy of 0.9961 has been achieved by this model which is relatively less than the transfer learning models that we implemented in Dataset-V2. It has a test loss

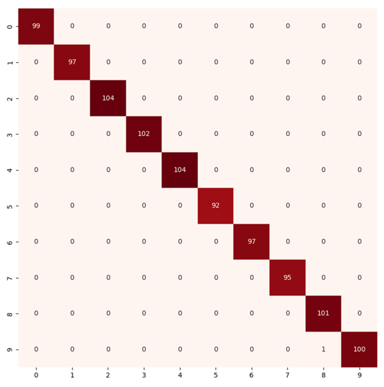

ResNet101

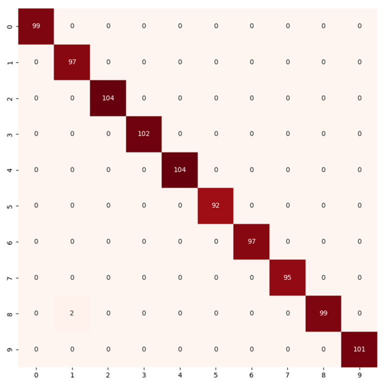

Xception

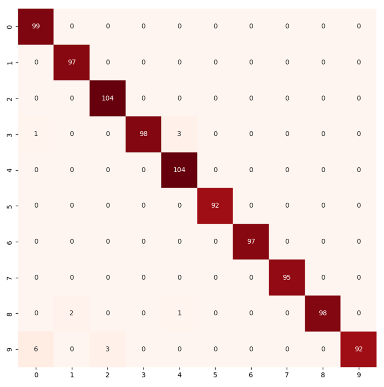

Inception-V3


InceptionResNetV2


DenseNet201


VGG16


MobileNetV2


3 Layer CNN

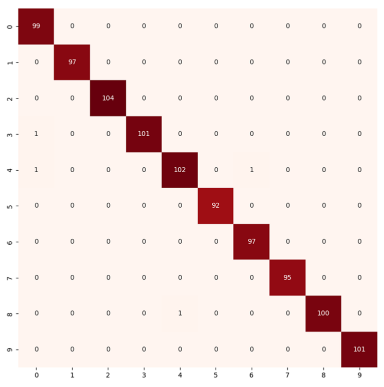

6 Layer CNN

Figure 6.1: Confusion Matrix of all models on Dataset-V2

of 0.0254 which is significantly higher than most transfer learning models. Therefore, even though the 6-layer CNN shows quite similar performance compared to the initially implemented 3-layer CNN while evaluating the test set of dataset, it still underperforms all of the transfer learning models except ResNet101.

From the figure 6.1 above, it can be seen that DenseNet201, VGG16, MobileNetV2 and both custom CNN models achieve the highest precision, recall and F1-score. The confusion matrix on smallest version of the dataset represents correct predicted and actual true outcomes and predicted-actual false outcomes as discussed in previous chapter. However, our confusion matrix is multi-dimensional consisting of 10 classes in total which also represents class-wise precision, recall and F1 score rates. Due to that we can understand the accuracy of model on unseen data through analysing the ratio of it successfully classifying among different classes.

6.1.2 Dataset Version - 1

After successful implementation of pre-trained models on Dataset-V2, those models were implemented on Dataset-V1, the medium sized dataset containing around one-fourth of the images of the original dataset and 3 times larger than Dataset-V2. Due to this, we can understand how these models perform with a slightly increasing dataset.

		Test Accuracy	Test Loss	Val(max) Accuracy	VAL(min) Loss	Avg Precision	Avg Recall	Avg F1 Score
Pre-Trained CNN Models	ResNet101	0.9736	0.1017	0.9708	0.1122	0.97	0.97	0.97
	Xception	0.9978	0.0171	0.998	0.0057	1	1	1
	InceptionV3	0.9945	0.0442	0.998	0.0096	0.99	0.99	0.99
	InceptionResNetV2	0.9978	0.0282	0.999	0.0071	1	1	1
	DenseNet201	0.9983	0.0132	1	0	1	1	1
	VGG16	0.9994	0.0061	0.999	0.0042	1	1	1
	MobileNetV2	0.9967	0.0264	0.999	0.0076	1	1	1
Custom CNN Models	CNN - 3 layer	0.9895	0.0345	0.9929	0.0268	0.99	0.99	0.99
	CNN - 6 layer	0.9945	0.0207	0.9959	0.0126	0.99	0.99	0.99

Table 6.2: Accuracy of Models in Dataset-V1

Pre-trained Models: Table 6.2 demonstrates that Resnet101 performs the weakest when assessing the model among all of the pre-trained CNN models. Similar to the results of Dataset-2 Resnet101 obtains the poorest test accuracy compared to other Transfer Learning models with an accuracy of 0.9736. InceptionV3 performs slightly better with an accuracy score of 0.9945. Mobilenet achieved 0.9967 test accuracy. Whereas, Xception and InceptionResNet-V2 show similar test accuracy of 0.9978 while DenseNet secures a higher accuracy of 0.9983. On the other hand, VGG16, with a 0.9994 test accuracy rate, outperforms all other Transfer Learning Models.

Custom CNN Models: Therefore, amidst the custom models mentioned on table 6.2 it can be noticed that 6-layer CNN achieved 0.9945 testing accuracy and test loss of .0207, showing a significantly better performance compared to the 3-layer CNN unlike observed in Dataset-V2.

In this version of dataset, Xception, InceptionResNetV2, DenseNet201, VGG16 and MobileNet-V2 and VGG16 achieves the best avg precision, recall and F1 score. The class-wise precision, recall and F1 score is given in the confusion matrix.

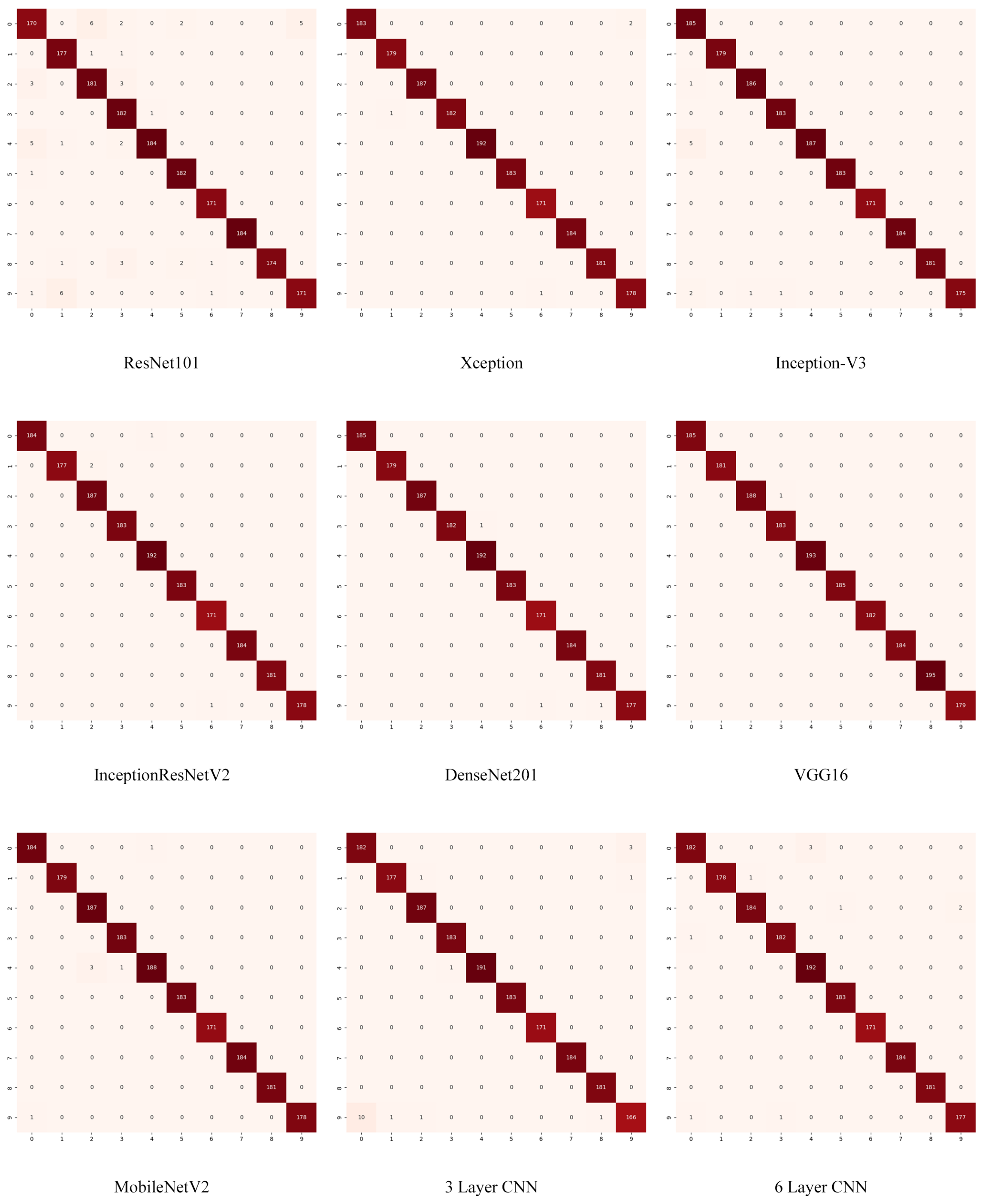


Figure 6.2: Confusion Matrix of all models on Dataset-V1

6.1.3 Original Dataset

This is the main version of dataset that has approximately 40 thousand images in total. Other versions of datasets were created by sub-setting this dataset in different ratios that has been mentioned in chapter 4. Due to being the largest dataset compared to other that has been dealt with for this study, this consumed more time in the training phase. Due to this, it has been trained until 30 epochs

		Test Accuracy	Test Loss	Val(max) Accuracy	VAL(min) Loss	Avg Precision	Avg Recall	Avg F1 Score
Pre-Trained CNN Models	ResNet101	0.9883	0.0408	0.989	0.0446	0.99	0.99	0.99
	Xception	0.9986	0.023	1	0	1	1	1
	InceptionV3	0.9988	0.0421	0.9989	0.0093	1	1	1
	InceptionResNetV2	0.9991	0.0123	0.9997	0.0028	1	1	1
	DenseNet201	0.9992	0.0219	1	0	1	1	1
	VGG16	0.9997	0.0013	0.9963	0.0001	1	1	1
	MobileNetV2	0.9998	0.0033	1	0	1	1	1
Custom CNN Models	CNN - 3 layer	0.9994	0.0028	0.9995	0.0025	1	1	1
	CNN - 6 layer	0.9986	0.00408	0.9995	0.0044	1	1	1

Table 6.3: Accuracy of Models in Original Dataset

Pre-trained CNN Models: From the above table 6.3, we can see that, the MobileNetV2 model has reached the highest accuracy with less training and validation loss. It is also noticeable that the accuracy of these models vastly vary due to their architecture while being implemented on the original dataset. VGG16 and MobileNetV2 show a splendid performance in terms of evaluating test data. Due to this, these two models are able to achieve the highest testing accuracy compared to all other pre-trained CNN models. Both of these models obtained very close testing accuracy as the difference of accuracy among VGG16 and MobileNetV2 is only 0.01%. Besides, DenseNet201 and Inception-ResNetV2 show outstanding results as well.

The difference between the actual and predicted outcomes, and a low loss number denotes that the classifier is more certain of its predictions which is measured by test loss. From the table it is observable that, despite achieving highest accuracy MobileNetV2's test loss is higher than the test loss obtained from the VGG16 model. However, the difference between the both models is very minimal.

Custom CNN: As seen in Table 6.3 above, 3-layer CNN model has attained an accuracy of 0.9994 on the test data of the original dataset, which outperforms most of the state of art transfer learning models unlike the results observed in both Dataset-V1 and V2. The test accuracy obtained by this model is significantly higher than the results obtained from training and evaluating ResNet101, Xception, InceptionResnet-V2 and DenseNet201 on the original dataset. However, it underperforms 2 transfer learning models, VGG16 and MobileNet-V2. 3-layer CNN model also has remarkably less test loss than that of the majority of transfer learning models. It obtains a test loss of 0.0028. On the other hand, our 6-layer CNN model fails to exceed the accuracy achieved by initially implemented 3-layer CNN model and most of the transfer learning models. It achieves test accuracy of 0.9986, similar to Xception. However, its training loss is notably less than most pre-trained CNN models.

In this main or largest dataset, all models except ResNet101 achieves the avg precision, recall and F1 score of 1. The class-wise precision, recall and F1 score is given bellow in figure 6.3

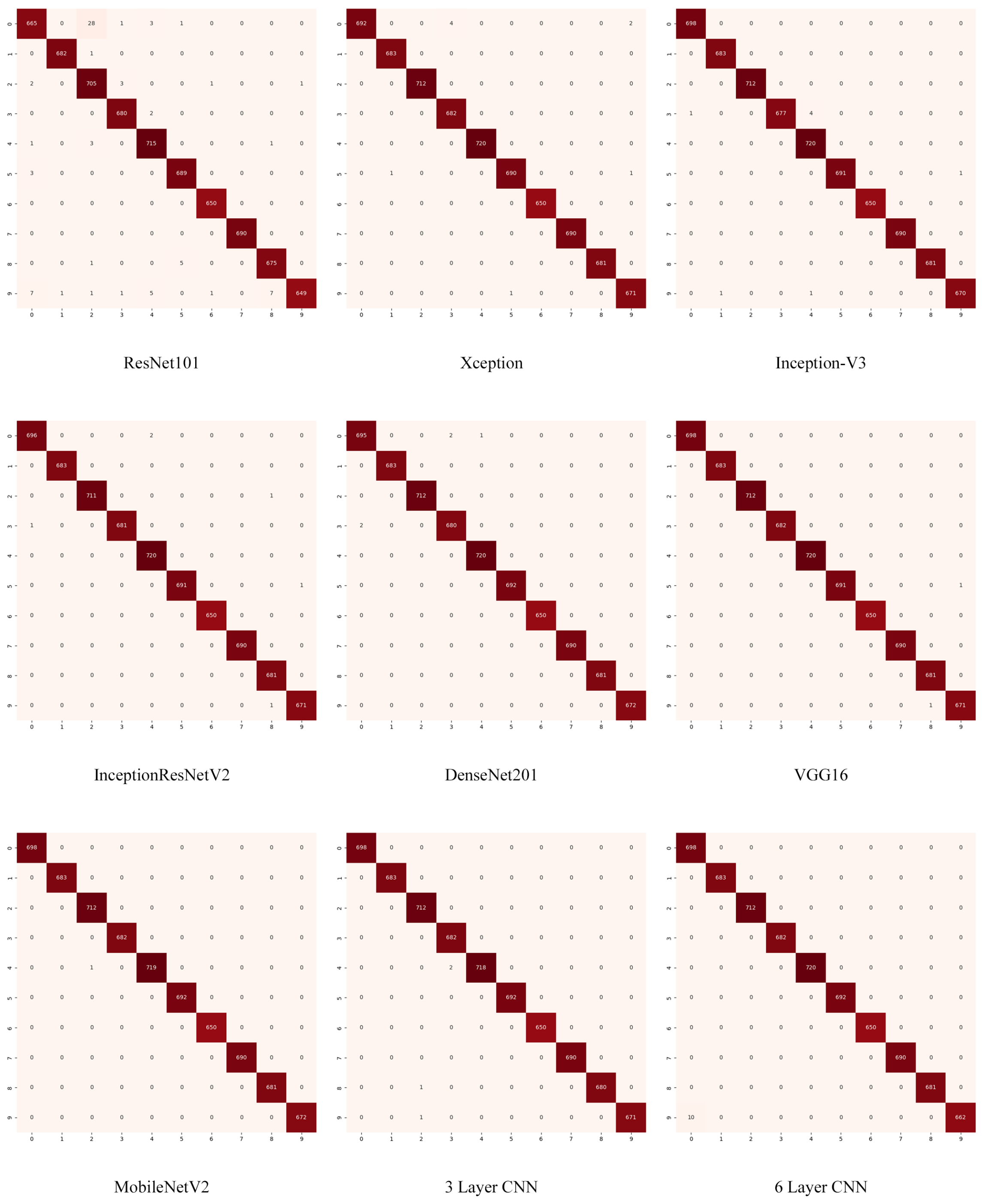


Figure 6.3: Confusion Matrix of all models on Original Dataset

6.2 Analyzing Training Accuracy-Loss Vs Validation Accuracy Loss

After thoroughly observing all of the pre-trained and custom CNN models through analyzing their performances on various versions of datasets based on several metrics, it can be concluded that VGG16 and MobileNetV2 performs the best, particularly considering the test accuracy and test loss achieved by both of these models.

Despite the fact that our custom CNN models perform outstanding by achieving great accuracy, those are unable to exceed the accuracy achieved by VGG16 and MobileNetV2. However, these alone do not ensure that the models' effectiveness as concerns regarding overfitting can also arise from high accuracy. This may cause poor results when the same models are applied on unseen data, and hence is undesired. In real life driving scenarios that our model is supposed to be implemented on, when the same models are applied on the unknown data, overfitting may lead to unsatisfactory outcomes. Due to this, both the pre-trained CNN and custom CNN models' training accuracy and loss were assessed to corresponding validation accuracy and loss for all three versions of datasets to ensure if the models truly perform well or the high accuracies are resulted from overfitting as illustrated by Figure 6.4, Figure 6.5 and Figure 6.6 beneath. Therefore, for the epoch acquiring highest validation accuracy has been considered as the best checkpoint result and saved. Later, the saved model was loaded while evaluating the model on unseen data which is test set of the dataset in our case. The test accuracy achieved from these models have been discussed in previous section

Dataset V2: As previously described in Chapter-2, presence of overfitting or underfitting issues can be ensured through observing the models' training and validation accuracy-loss graphs generated after each model has been fed into with training data. As our model performs well in training phase, it can be said that there is no issue of underfitting so far, hence to spot any case of overfitting among these models training vs validation accuracy and loss have been observed. If mentioned previously, if the training and validation accuracy or loss curves have significant difference among that can be considered as a case of overfitting.

While observing the graphs of all models generated after being trained on the smallest version of dataset, Dataset-V2 for 100 epochs, we can see that most of the models' graphs doesn't have much difference among training-validation loss curves and train-validation accuracy curves. However, ResNet101, Xception and Inception-V3 shows hints overfitting in the very first few epochs.

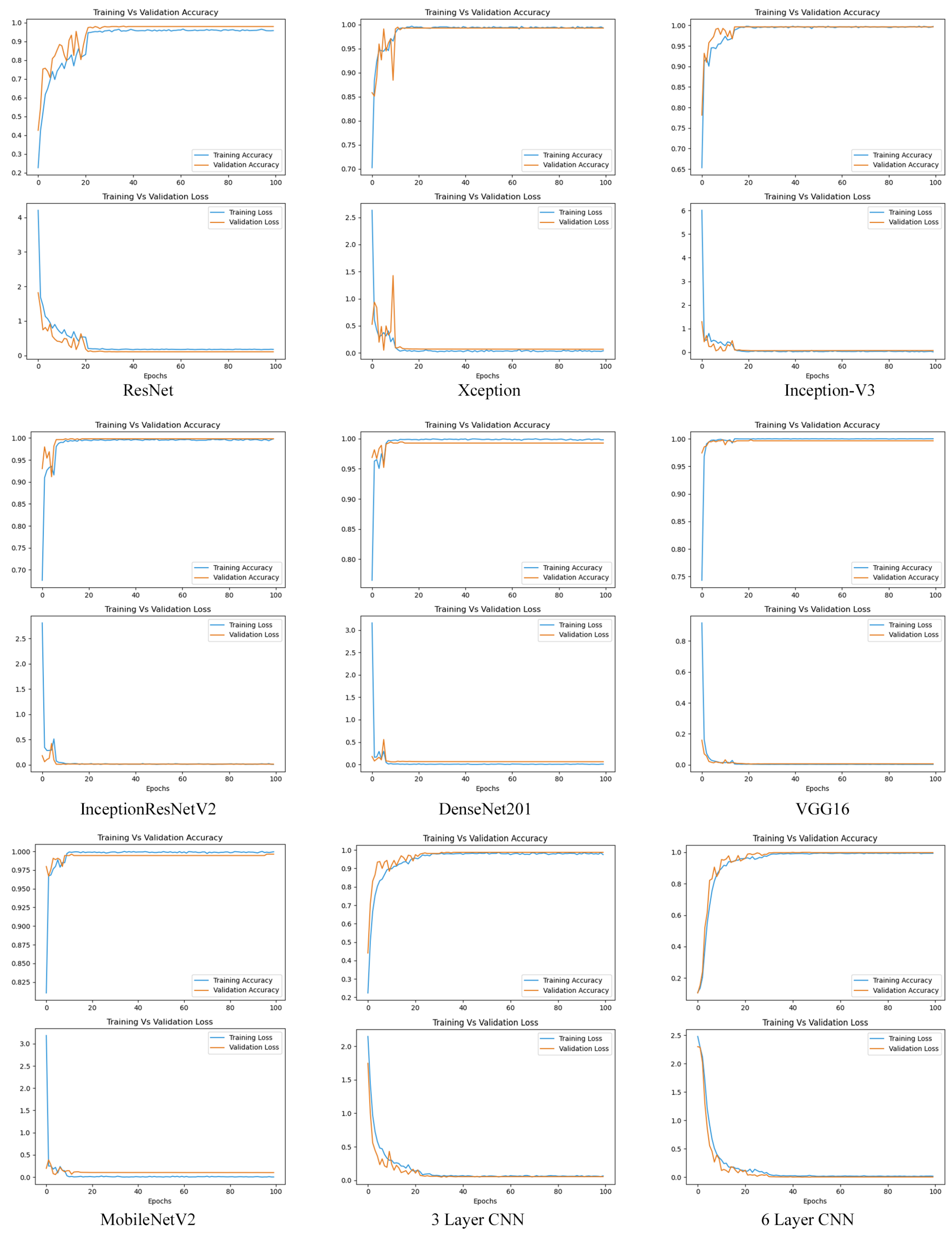


Figure 6.4: Training Vs Validation Accuracy-Loss Graphs of Implemented Models On Dataset-V2

Dataset-V1: This version of dataset is larger than the Dataset-V2. For this version the dataset has been trained until 70 epochs.

After monitoring the graphs of Figure 6.5, it can be noticed that, in case of ResNet101, at the initial epochs there are slight differences among both training-validation accuracy and loss which ultimately gets closer. For the other pre-trained model a sudden

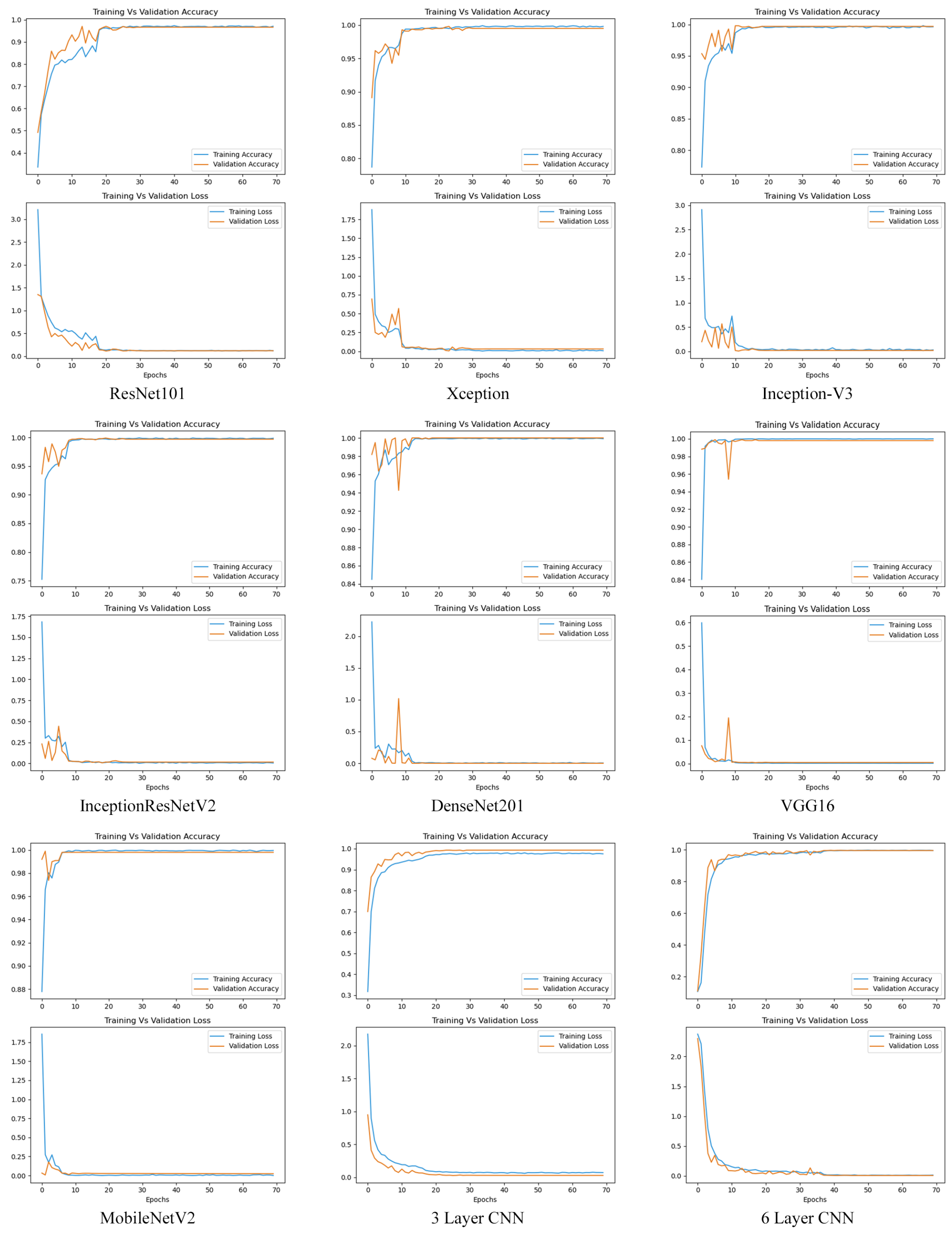
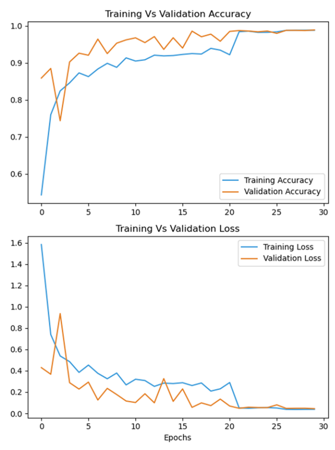


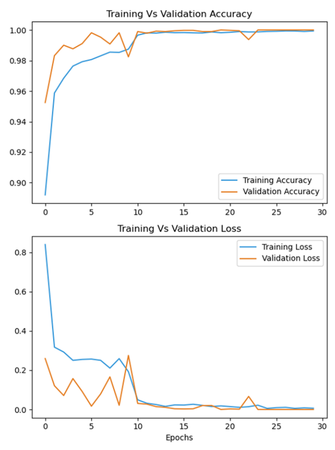
Figure 6.5: Training Vs Validation Accuracy-Loss Graphs of Implemented Models On Dataset-V1

rise of validation loss can be seen. However, this is natural while training a model and cannot be considered as an issue of overfitting, as the best epoch has been saved and applied in the model. And since that epoch is the highest from where training and validation curve is alligned there is no case of overfitting. Nonetheless, in terms of MobileNetV2, 3-Layer CNN and 6-Layer CNN models the curves of validation training loss and accuracy completely converges with each other showing almost no chances of the model being overfitted while training.

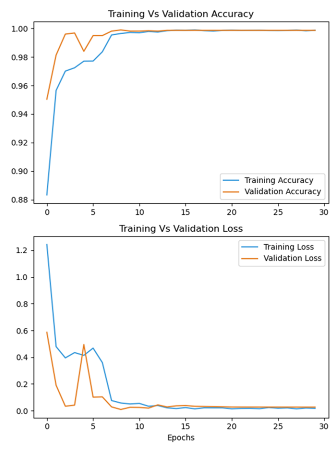
Original Dataset: In original dataset as well it can be seen that no model shows any chances of being overfitted.



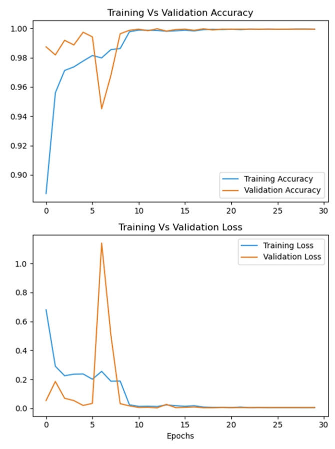
ResNet101



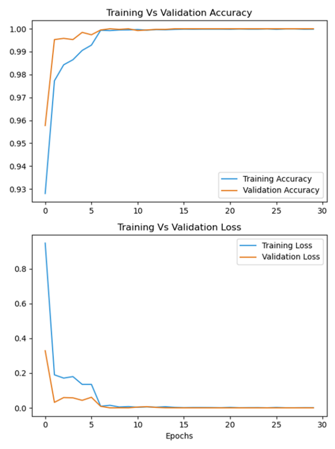
Xception



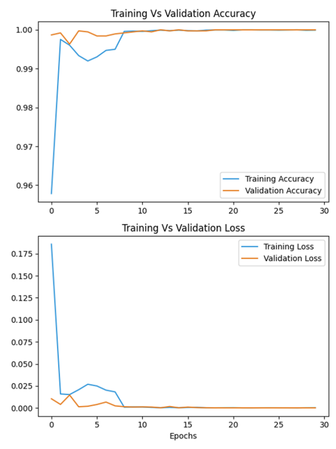
Inception-V3



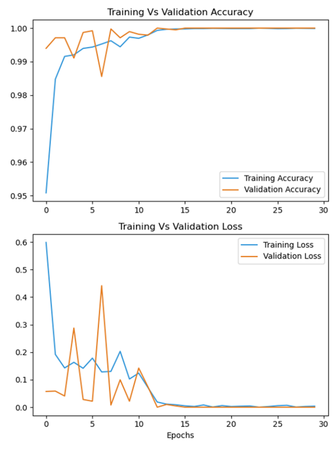
InceptionResNetV2



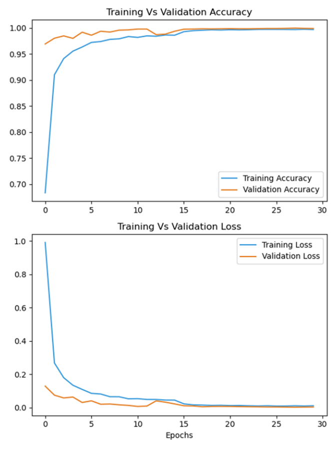
DenseNet201



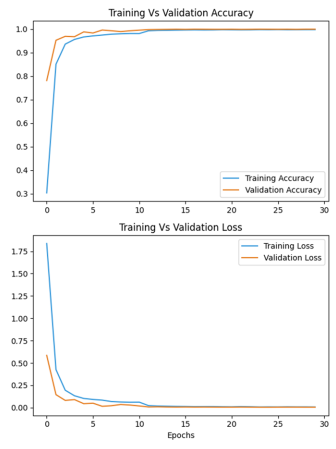
VGG16



MobileNetV2



3 Layer CNN



6 Layer CNN

Figure 6.6: Training Vs Validation Accuracy-Loss Graphs of Implemented Models On Original Dataset

Therefore, similar to Dataset-V2 and Dataset-V1, both of the custom CNN models are showing phenomenal performance with no overfitting issues unlike the transfer learning models that showed considerably smoother in the smaller versions of the dataset.

After observing the training and validation accuracy-loss graphs of several models of different versions of dataset, it can be seen that almost all of the implemented models are giving us an impressive performance making it hard to decide on one model that is performing the best. Therefore, in terms of test accuracy, InceptionV3, Xception, DenseNet201, VGG16, MobileNetV2 and 3 layer CNN performs quite better than 6-layer model. However, according to our observation in training-validation loss and accuracy graph, VGG16, MobileNetV2 and both custom CNN models, 3-layer CNN and 6-layer CNN has performed significantly better than other models.

6.3 Parameters Of Models

In the case of deep learning, parameters include weight and bias, which are characteristics of the training data that will be learned throughout the learning process. The summation of all the biases and weights of the neural network indicates the total number of parameters. If the history of deep learning is observed, we can see that the number of parameters has resulted in appreciably good outcomes. Deep learning models, however, need a lot of computational resources, so they're not ideal for lightweight devices like smartphones and the Internet of Things (IoT). The amount of parameters caused due to the architecture of a specific model has a lot to contribute in efficient detection as high parameters often result to high computational and time complexity. As the focus of this dissertation is to detect several distractions of drivers caused in real-time scenarios while setting a camera device on a side of the vehicle, the time required to perform the detection by the algorithm is an important factor. Due to this, the detection procedure in such crucial real-time based applications should take the least amount of time and to do so the goal should be minimizing the time and computational complexity to the minimum.

Model	**Parameters**
ResNet101	43,661,706
Xception	21,865,010
InceptionV3	22,314,794
InceptionResNetV2	54,720,746
DenseNet201	19,262,794
VGG16	14,965,578
MobileNetV2	3,730,634
3-Layer CNN	508,298
6-Layer CNN	1,793,514

Table 6.4: Parameters Of Implemented Models

Among all of the models that has been implemented in this work, whether pre-trained or custom CNN, all has shown great training and testing accuracy with minimal to almost no issues of overvitting. However, based on training and testing accuracy and loss among pre-trained CNN models' VGG16 and MobileNetV2 models are the best performer for our task. On the other hand, from custom both CNN models' 3 layer CNN outperforms the 6- layer CNN model in terms of accuracies.

Nevertheless, judging from the perspective of time complexity and the amount of parameters required for these models, VGG16 and MobileNetV2 has least amount of parameters compared to other transfer learning models, where parameter of MobileNetV2 is around 3.73 million which is four times less than the parameters of VGG16. Therefore, our custom CNN models, 6-layer CNN and 3-Layer CNN has parameters of around 1.7 million and 0.5 million which is approximately 2.08 and 7.34 times less than MobileNetV2.

So judging not only from the constraints of accuracies but also the parameters, considering these models' implementation in real-time, among all of these 9 implemented models MobileNetV2, 3 Layer CNN and 6 layer CNN models are best suited for this field.

6.4 Interpretability Of Model

Because of the availability of data, AI systems have widespread in numerous different fields and been implemented in several applications these days, which has been possible solely due to the advancements of Deep Learning and the emergence of various creative ways alongside, through which these growing datas can be utilized. Developments in the sector of Deep Learning has allowed us to construct models that are able to perform well on increasingly complex tasks. However, as a consequence, the complexity of these systems and applications has become incomprehensible to visualize. Due the increasing parameters and complexity of these models, it is quite challenging to understand how models arrive at their predictions, which again makes it hard for us to analyze when a model is resulting poor performance, for exact which reasons the model might not be working well enough. Due to this, machine learning and deep learning models are often referred as black boxes.

However, deep learning is also implemented in various safely and health criticle environments such as healthcare sector or autonomous driving. Therefore, human simply need to understand the perspectives of the model through which they learn and evaluate, to build a trust among human and machines even in crucial sectors. That's when we are required to know and explain the model's learnings. The field of Interpretable Machine Learning works on better understanding the interpretability of these neural network models and validate how this deep learning models work.

There are several techniques that approximates black box models. For example, LIME, Gradcam, SHAP are most reknowned ones for such tasks. To check interpretability of our model we have used LIME.

The goal of checking interpretability of implemented models using LIME was to understand if these models are properly detecting the Regions of Interest(ROI) that these are supposed to look into. For example, if the input image for testing belongs to c1 class which is texting with right hand, we would expect our model to identify either the hand or phone or both. LIME aims to comprehend the characteristics that affect a black-box model's prediction of a particular ROI. The list of explanations that LIME produces demonstrates the significance of each feature value to prediction of model.

6.4.1 Interpretability of 3-Layer CNN

At first the we checked explainability in 3-Layer CNN model. This is the custom CNN model that achieved the highest accuracy. Figure 6.7 below shows the region of interest according to the confidence value and marks the highest ROI that plays the most crucial role in the prediction of the model for that certain class. Besides, through the heatmap most influential segments used by the model to make the prediction can be identified. To check interpretation of these models it has been fed into some images where each of those belong from 10 different classes of our dataset.

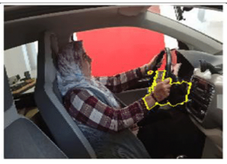
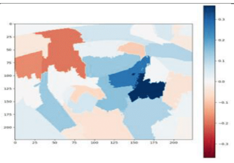
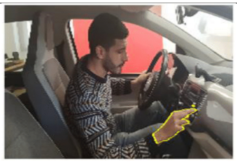
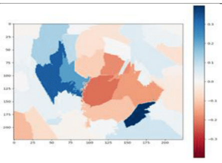
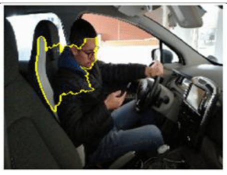
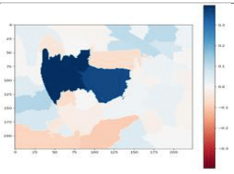
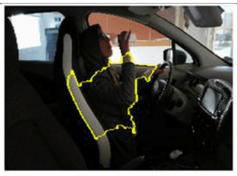
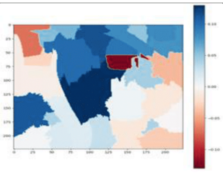
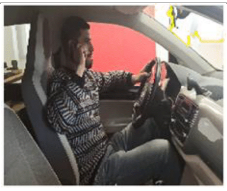
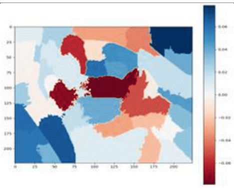
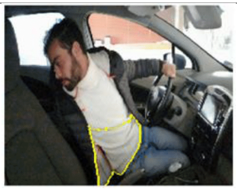
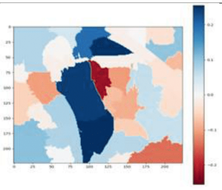
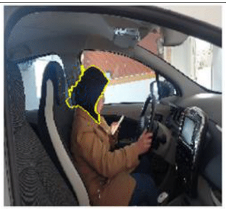
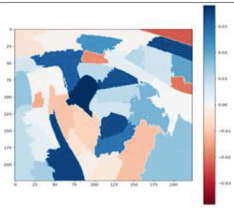
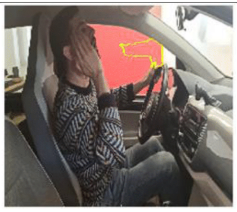
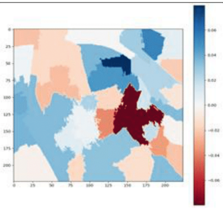
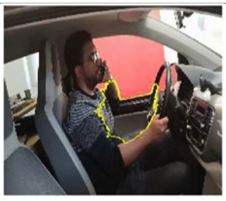
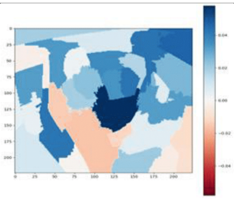
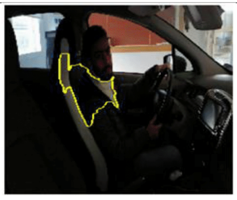
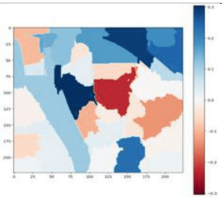
Class	Identified Region	Heat Map	Class	Identified Region	Heat Map
C0: Safe Driving			C5: Operating the radio		
C1: Texting-Right hand			C6: Drinking		
C2: Talking on the phone-Right hand			C7: Reaching behind		
C3: Texting-Left hand			C8: Tidying up hair or applying makeup		
C4: Taking on the phone-Left hand			C9: Talking to passenger		

Figure 6.7: ROI Identified For 3 Layer CNN

Figure 6.7 containes 10 images from 10 classes. It can be noticed that, our 3 layer CNN model is successfully able to identify regions of interests for class C0, C4, C5, C6, C7, C9. For C0 class it identifies the hand that is on steering based on the information the image can be recognized as an image of "Safe Driving" class. Whereas, in C4, the elbow of left hand is identifies as the person is talking over phone using his left hand. For C5 class, marked hands on radio region shows that the driver is busy operating the radio of the vehicle. In terms of C6, C7 and C9 class, drivers' elbow, waist and shoulder is identified as most prominent region with the most confidence value.

However, our 3 layer custom CNN model fails to identify the proper regions for class C2 and C8. Therefore, even though it identifies certain postures for C1 and C3 class, it can also be considered as wrong regions. For example, in C1 class the model marks driver's tilted head as most significant region where the model actually should rather focus on right hand and phone as the driver is texting with his right hand. Same can be seen for C3 where the driver is using phone with left hand but the model neither identifies the left hand or phone rather is again focused on tilted head. In brief, the custom layer CNN model is able to identify 6 classes properly and identifies 4 regions improperly.

6.4.2 Interpretability of MobileNetV2

This model has shown us the best performance among all of the implemented models with best accuracy. Also it has the least parameters among those, making it more suitable than the other ones for real time application.
Figure 6.8 below displays the region of interest based on the confidence value and designates the highest ROI that is particularly crucial for the MobileNetV2 model's estimation for that given category.

If the figure 6.8 is observed thoroughly it can be seen that, the model perfectly identifies C1, C4, C5, C6, C9. where for C1 it marks right hand with phone, for C4 it sees left hand with phone and face, , for C5 the model spots right hand operating radio, for C6 it marks the water bottle and hand and lastly for C9 it spots the face of the driver. However, the model fails to identify proper region for C0, C2, C3, C7 and C8.

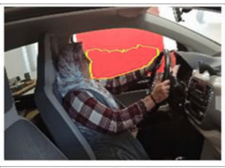
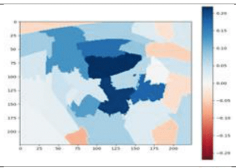
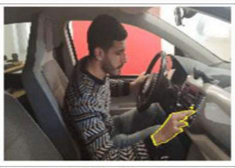
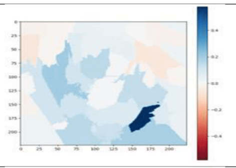
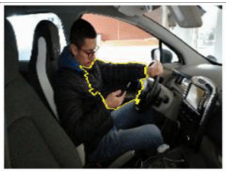
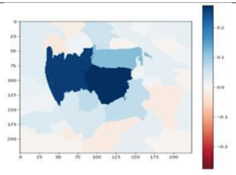
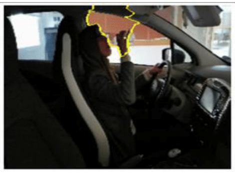
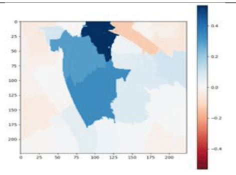
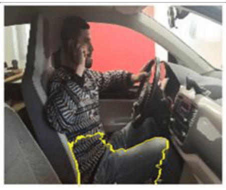
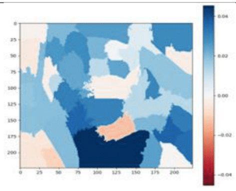
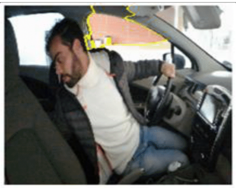
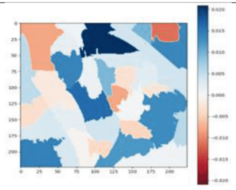
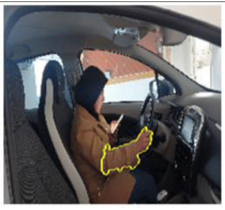
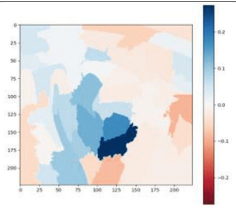
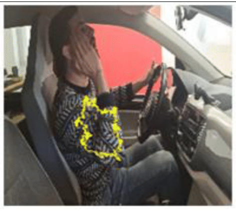
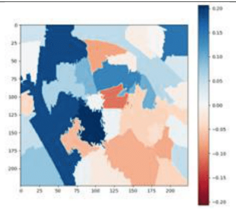
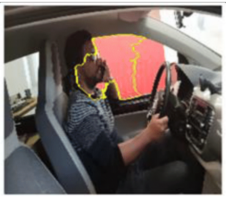
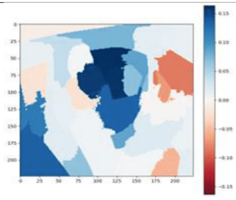
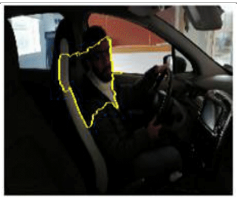
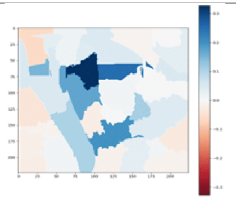
Class	Identified Region	Heat Map	Class	Identified Region	Heat Map
C0: Safe Driving			C5: Operating the radio		
C1: Texting-Right hand			C6: Drinking		
C2: Talking on the phone-Right hand			C7: Reaching behind		
C3: Texting-Left hand			C8: Tidying up hair or applying makeup		
C4: Taking on the phone-Left hand			C9: Talking to passenger		

Figure 6.8: ROI Identified For MobileNetV2

6.4.3 Interpretability of 6 Layer CNN

While analyzing the results of every models it was seen that 6-layer CNN performs worse than other models in terms of test accuracy and loss. It's total parameter is approximately 1.7 million which is a lot less than transfer learning models. Figure 6.9 shows the the estimated regions with highest significance by this model.

Here we can see that, unlike MobileNetV2 and 3-Layer CNN, 6 layer CNN model can identify proper regions of interests of all the classes quite precisely. If we try to explore all of the classes individually we will notice that, in c0(safe driving) the model identifies hands on steering as the most important region. For C1(Texting-right hand) it exactly points out the right hand along with the device. In case of C2(Talking on phone-Right hand) our model precisely identifies right hand and phone and in C3(Texting-left hand) it can identify phone and left hand. Therefore, observing C4(Talking on phone- left hand) the model identifies the driver's left elbow and the hand along with certain finger gets identified in terms of C5(Operating radio). In C6(Drinking), the model identifies not only uplifted head but also misplaced right elbow as next prominent region for this certain class. In the matter of C7(Looking behind) and C8(Fixing hair & makeup) the model accurately identifies the front portion of body and hand. Lastly our model correctly spots the significant

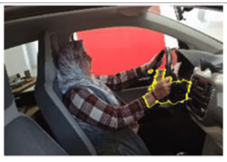
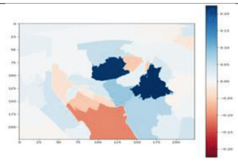
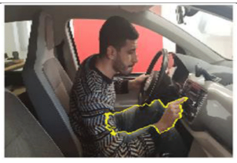
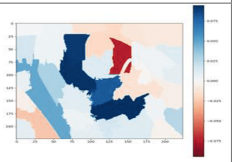
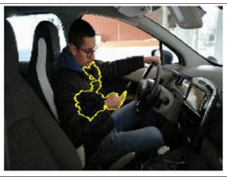
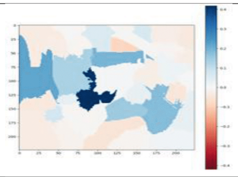
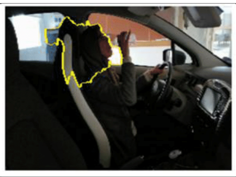
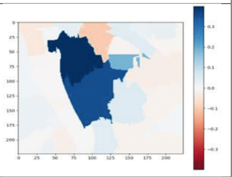
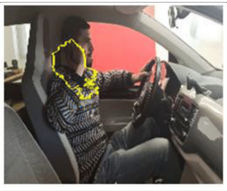
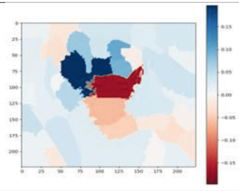
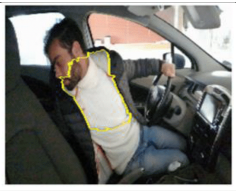
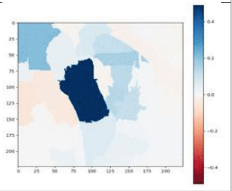
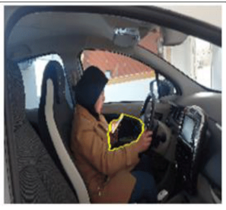
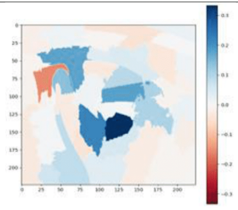
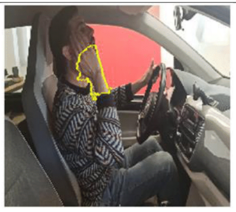
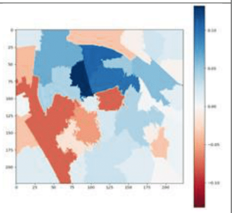
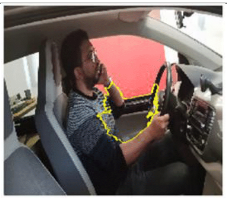
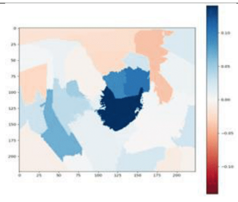
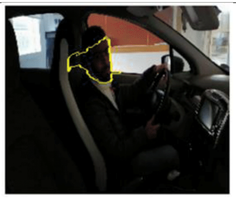
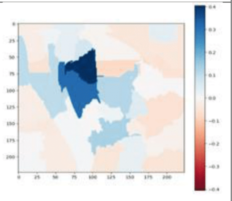
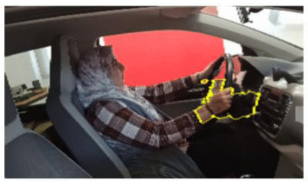
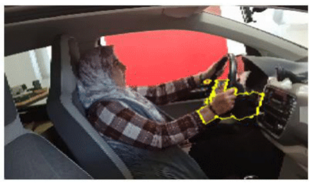
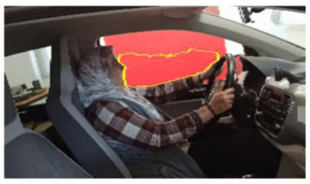
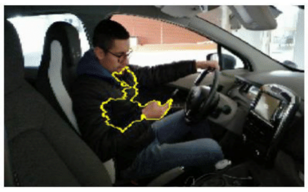
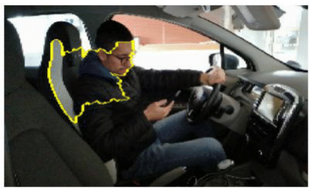
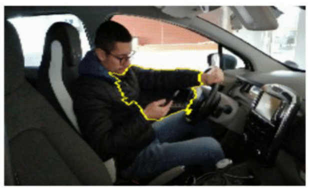
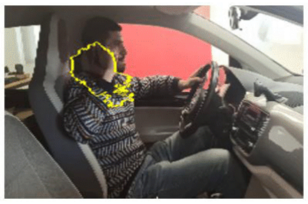
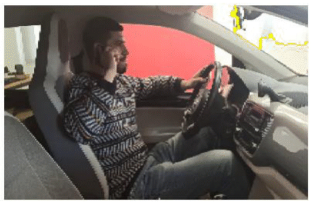
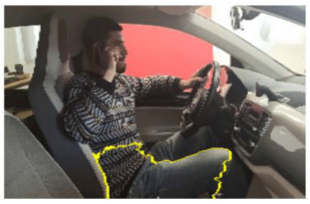
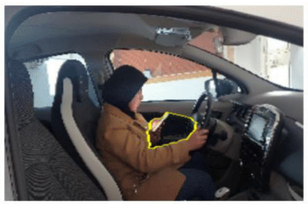
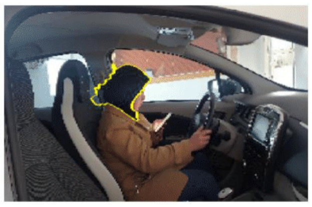
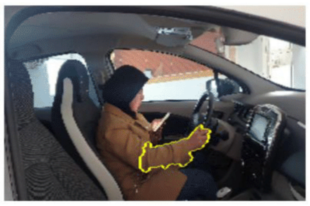
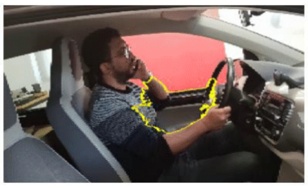
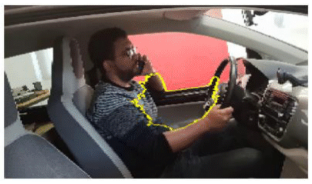
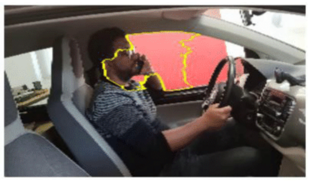
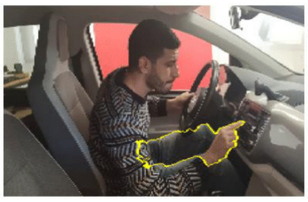
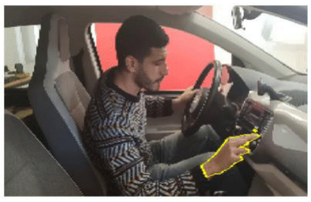
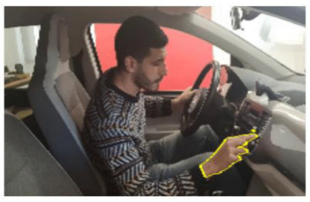
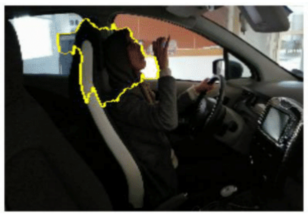
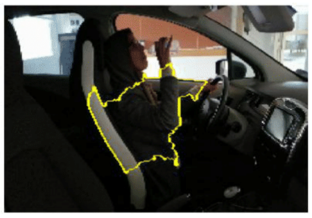
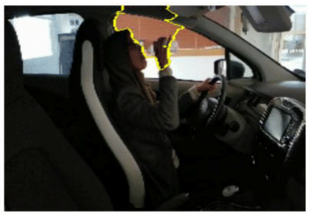
Class	Identified Region	Heat Map	Class	Identified Region	Heat Map
C0: Safe Driving			C5: Operating the radio		
C1: Texting-Right hand			C6: Drinking		
C2: Talking on the phone-Right hand			C7: Reaching behind		
C3: Texting-Left hand			C8: Tidying up hair or applying makeup		
C4: Taking on the phone-Left hand			C9: Talking to passenger		

Figure 6.9: ROI Identified For 6 Layer CNN Model

region of C9(Talking to passenger) as it spots the face.

6.4.4 Comparison

Here, figure 6.10 shows the comparison of ROI identified by our proposed CNN model(6 layer CNN model), Custom CNN model(6 layer CNN model) and transfer learning model MobileNetV2. To do a fair evaluation while doing the comparison among all these three models it has been ensured that same image of the test set of each class has been fed into the model to identify the region of interest. Therefore we can see that our proposed model performs best as it is able to identify ROI better than 3 layer CNN and MobileNetV2.

Class	Proposed CNN Model Parameters : 1.7M	Custom CNN Model Parameters: .5M	MobilenetV2 Parameters: 3.4M
C0: Safe driving			
C1: Texting-Right hand			
C2: Talking on the phone-Right hand			
C3: Texting-Left hand			
C4: Talking on the phone-Left hand			
C5: Operating the radio			
C6: Drinking			

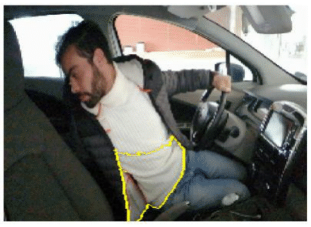
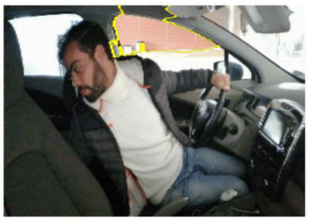
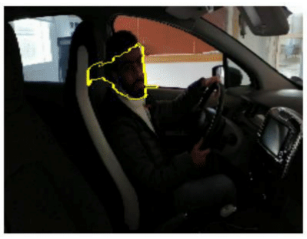
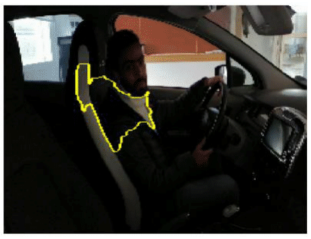
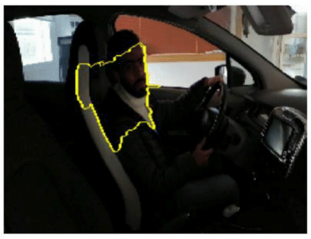
C7: Reaching behind			
C8: Tidying up hair or applying makeup			
C9: Talking to passenger			

Figure 6.10: Comparision Among Models

6.5 Proposed CNN Model Result In SFD3 Dataset

After scrutinizing our proposed model, 6 layered custom CNN's excellent performance in Howdrive: Distracted driver dataset, we implemented our model on famous State farm distracted driver dataset (SFD3). Therefore the achieved test accuracy of the model in State farm distracted driver dataset is 99.20% and test loss is as low as 0.0332 as can be seen from the test accuracy and loss graph as can be noticed from figure 6.11.

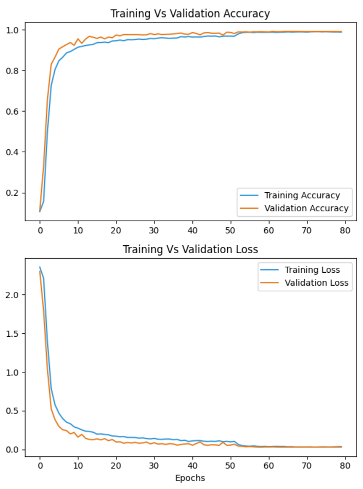
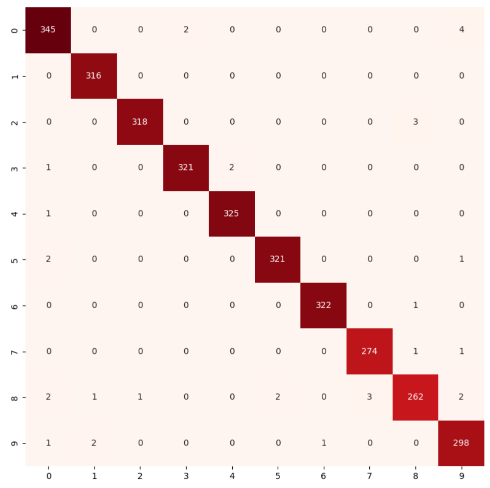


Figure 6.11: 6 Layer CNN Validation-Training Accuracy Loss Graph In SFD3 Dataset

Therefore an average precision, recall and F1-score of 99% has been achieved by our model. The confusion matrix of figure 6.12 shows class wise result of our 6 layer CNN model in SFD3 dataset.



	0	1	2	3	4	5	6	7	8	9
0	345	0	0	2	0	0	0	0	0	4
1	0	316	0	0	0	0	0	0	0	0
2	0	0	318	0	0	0	0	0	3	0
3	1	0	0	321	2	0	0	0	0	0
4	1	0	0	0	325	0	0	0	0	0
5	2	0	0	0	0	321	0	0	0	1
6	0	0	0	0	0	0	322	0	1	0
7	0	0	0	0	0	0	0	274	1	1
8	2	1	1	0	0	2	0	3	262	2
9	1	2	0	0	0	0	1	0	0	298

Figure 6.12: 6 layer CNN Confusion Matrix In SFD3 Dataset

We also analyzed if our model is able to detect region of interests as impressively in other datasets. Regions of all 10 classes has been identified perfectly as it was seen in previously implemented Howdrive distracted driver dataset. The figure 6.13 shows the results.

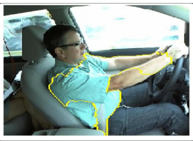
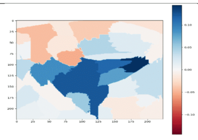
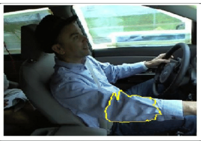
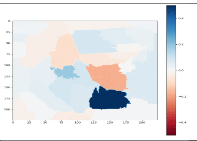
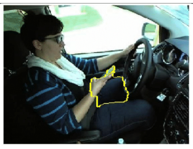
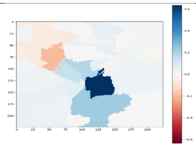
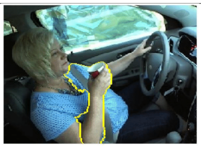
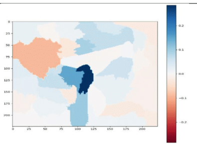
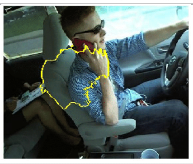
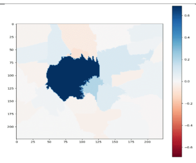
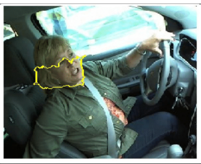
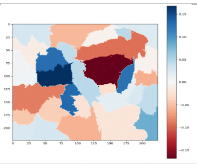
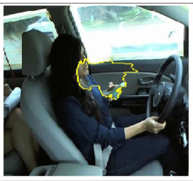
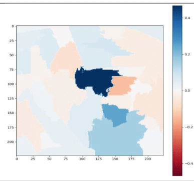
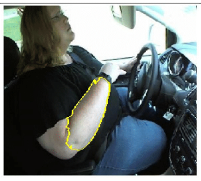
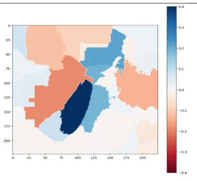
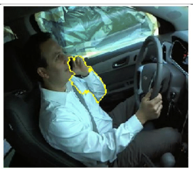
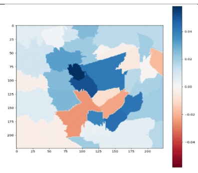
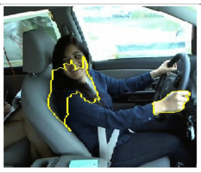
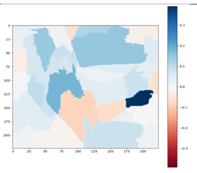
Class	Identified Region	Heat Map	Class	Identified Region	Heat Map
C0: Normal driving			C5: Operating radio		
C1: Texting - Right			C6: Drinking		
C2: Talking on phone- Right			C7: Reaching Behind		
C3: Texting Left			C8: Hair and Makeup		
C4: Talking On Phone - Left			C9: Talking To Passenger		

Figure 6.13: 6 layer CNN ROI Identification in SFD3 Dataset

6.6 Comparison of Proposed Model With Previous Related Works

Throughout the years, many researches have been done in this field and researchers and scientists have progressed to detect distracted behaviour with greater accuracy. Table 6.5 shows the models that has been used in those previous works and the accuracy that has been obtained by those models. This has also been broadly discussed Chapter 3 of our paper.

Paper Title	Applied Model	Accuracy
Distracted Driver Detection and Classification [11]	Average of pre-trained VGG_16, VGG_19 and Inception V3	96.31%
Detection of Distracted Driver using Convolutional Neural Network [10]	VGG with Regularization	96.31%
Distracted Driver Detection [17])	VGG-16 architecture weights and applied the Dense and Flatten layers.	93%
An Efficient Deep Learning Framework for Distracted Driver Detection [22]	EfficientDet-D3	99.16%
Driver Distraction Identification with an Ensemble of Convolutional Neural Networks [13]	Weighted sum of pre-trained AlexNet, InceptionV3 , a ResNet50, and a VGG-16	90%
HSDDD: A Hybrid Scheme for the Detection of Distracted Driving through Fusion of Deep Learning and Handcrafted Features [25]	KNN Algorithm	95.9%
An Embedded Deep Learning Computer Vision Method for Driver Distraction Detection [21]	Pre-trained SqueezeNet	93%
A hybrid neural network for driving behavior risk prediction based on distracted driving behavior data [27]	Custom network named Driving Behavior Risk Prediction Neural Network (DBRPNN)	91.46%
Robust Deep Learning- -Based Driver Distraction Detection and Classification [19]	VGG_19	95.77%
Distracted Driving Detection by Combining ViT and CNN [28]	Combination of ViT and KNN	97.9%
Application of CNN-based architectures in Detection of Distracted Diver	**Custom 6_Layer CNN**	**99.86%**

Table 6.5: Comparing our work with previous related works

From the table we can clearly see that our model is able to surpass their accuracy. However, as we have mentioned several times before, the objective of this paper has never been gaining a higher accuracy but the gaining better identification of distracted behavior. None of these previous work has explained their model. Though the accuracy of these works are quite impressive but, higher accuracy certainly do not ensure proper detection of ROI as we have previously seen from MobileNetV2 and 3-layer custom CNN.

Chapter 7

Discussion

After constructing several convolutional neural network models, our 6 Layer CNN model appeared to be the most suitable system that accomplishes our goal of providing an efficient system which is able to classify several distracted driving behaviours with better success rate. Not only our proposed model has impressive test accuracy and low loss along with high precision, recall and F1 score, it also overcomes the phenomenon of a model being overfitted. It does, however, raise an interesting concern regarding the model's capability to triumph over the previously built state-of-art, high-level pre-trained models.

Even though our model has not been able to achieve a greater accuracy than pre-trained models, it surely shows a better interpretability results with minimal parameters compared to those. It is quite difficult to know what specific reasons triggered our model to perform better than transfer learning models, however, one particular reason may be deduced from the research process. As we know, deep learning models are known to be "Data hungry", which means that, to train and make a model learn requires sufficient amount of data, where those datas must be well labeled. Transfer learning models are trained on ImageNet dataset, a large visual database containing roughly 14 million images on 20 thousand different classes. To be trained on this huge database, these models were required to show impressive accuracy with least error rate while learning datas of numerous sources and categories and to do so these transfer learning model needed more layers and parameters. Moreover, if we notice the evolution of deep learning models starting from AlexNet, we will see that the error rate of these evolving transfer learning models tend to be reduced with increasing numbers of layers, in other words with increasing complexity. We are assuming that, due to the previously learned weights from images of different subjects, those pre-trained models have the tendency to search for distinct patterns that are unsuitable for our images. For example, in terms of c0(Safe driving) and c7(Looking behind), one of the transfer learning models with best performance, MobileNetV2 was seen emphasizing on car window instead of identifying distinct human body parts particularly involved for classification of these classes. This issue doesn't arise in our case as our proposed model is specifically trained to perform this certain classification while learning from the images of training set of data. The dataset that we used for our research purpose has around 40 thousand images where 28.8 thousand images were used specifically to training our model. This way our model extracts only the exact required informations from each convolution layer and proceed it to the next layer, learning what it is supposed to look at particularly without dealing with these unnecessary extra layers and calculations due to previously stored weights.

There have been some related researches in this field that has achieved impressive accuracy. However, as we have seen from our previous analysis in this study, accuracy should not be the only metric to evaluate a model's performance. Even though MobileNetV2 and 3 layer CNN which had even lesser parameters (0.5 million) achieved higher accuracy than our proposed model, we saw that those were not able to identify all classes due to logical reasons. Which means, in real life scenarios these models will not be able to perform as efficiently. However, in crucial sectors as such it is important to build trust within technology and humans. Due to this reason, it is necessary to know what these models are learning to make a successful practical implementation possible. However, no work has been done before that ensures this.

Although it could appear that the study and its findings were flawless, these are still not prone to limitations. Even though our model shows great potential, we have some limitations as well. Firstly, we haven't still evaluated our model within completely different unseen data which it will need to do in real-life scenarios. Secondly, as our model is supposed to work on real-time application where videos will be extracted to frames which again will go through all the steps mentioned in this work, it is necessary to apply this in such way to see if our model is able to identify distracted behaviours within required short amount of time. We will be working on this further in future and implement our research as a project based model to implement in reality.

Chapter 8

Conclusion

Distraction of any form is not good. And when it comes to driving it can be lethal. It can put ourselves and our loved one's life on the line. A message pops up while driving and it might seem harmless to check and reply that text, but little do we think about the damage it can cause to our life. People face fines, tickets and even jailtime for such offense. But the biggest of them all, we might have to trade our life because of these distractions while driving.

With the goal of outspreading positive impacts by minimizing such occurrences, after looking into several CNN application based approach, we are proposing a custom 6 layer CNN model that is able to identify the common behaviours attempted by drivers that lead to distraction while driving. After finding the suitable dataset the dataset has been properly augmented and several data pre-processing has been done. Later on, after implementing our model in several pre-trained models and analyzing an convolutional approach has been taken to build a suitable custom model for this task. Later on after trying out several custom CNN models 2 models, 3-layer custom CNN and 6-layer custom CNN that worked best has been chosen to be portrayed in this work. Our model has been built and trained specifically to work on this domain with impressive accuracy and low loss. Our proposed model achieves accuracy of 99.86% which surpasses accuracies obtained by most other related works. Moreover, the model is able to explain how it signifies the regions and this prove that the proper information and features were extracted through every convolution layers which ultimately leads to proper identification. Later on, the proposed 6-layer CNN model has been implemented on another famous dataset, State farm distracted driver dataset of Kaggle and there as well it achieves a splendid accuracy of 99.20% and alongside our model shows great explinability with correct region identification in terms of this dataset as well.

Bibliography

[1] Y. Liang, *Detecting driver distraction.* The University of Iowa, 2009.

[2] W. H. Organization *et al.*, “Mobile phone use: A growing problem of driver distraction,” 2011.

[3] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv preprint arXiv:1207.0580*, 2012.

[4] F. Tango and M. Botta, “Real-time detection system of driver distraction using machine learning,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 894–905, 2013.

[5] R. A. Berri, A. G. Silva, R. S. Parpinelli, E. Girardi, and R. Arthur, “A pattern recognition system for detecting use of mobile phones while driving,” in *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, IEEE, vol. 2, 2014, pp. 411–418.

[6] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[7] *State farm distracted driver detection*, 2016. [Online]. Available: https://www.kaggle.com/c/state-farm-distracted-driver-detection/data.

[8] Y. Abouelnaga, H. M. Eraqi, and M. N. Moustafa, *Auc distracted driver dataset*, Oct. 2017. [Online]. Available: https://abouelnaga.io/projects/auc-distracted-driver-dataset/.

[9] A. Aksjonov, P. Nedoma, V. Vodovozov, E. Petlenkov, and M. Herrmann, “Detection and evaluation of driver distraction using machine learning and fuzzy logic,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 6, pp. 2048–2059, 2018.

[10] B. Baheti, S. Gajre, and S. Talbar, “Proceedings of the ieee conference on computer vision and pattern recognition workshops,” 2018, pp. 1032–1038.

[11] P. M. Chawan, S. Satardekar, D. Shah, R. Badugu, and A. Pawar, “Distracted driver detection and classification,” *International Journal of Engineering Research and Applications*, vol. 4, no. 7, 2018.

[12] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: An overview and application in radiology,” *Insights into imaging*, vol. 9, no. 4, pp. 611–629, 2018.

[13] H. M. Eraqi, Y. Abouelnaga, M. H. Saad, and M. N. Moustafa, “Driver distraction identification with an ensemble of convolutional neural networks,” *Journal of Advanced Transportation*, vol. 2019, 2019.

[14] S. Kusuma, J. D. Udayan, and A. Sachdeva, "Driver distraction detection using deep learning and computer vision," in *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, IEEE, vol. 1, 2019, pp. 289–292.

[15] S. Jeong-ro, *State farm distracted driver detection*, Nov. 2020. [Online]. Available: https://www.kaggle.com/datasets/rightway11/state-farm-distracted-driver-detection.

[16] P. Mao, K. Zhang, and D. Liang, "Driver distraction behavior detection method based on deep learning," in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 782, 2020, p. 022 012.

[17] D. Ruparel, A. Rajde, S. Shah, and P. Gidwani, *Distracted driver detection*, Oct. 2020. [Online]. Available: https://www.jetir.org/view?paper=JETIR2010371.

[18] D. J. K. Amy Schick Debbie Ascone, "Distraction by cell phones and texting," *National Highway Traffic Safety Administration*, 2021.

[19] A. Ezzouhri, Z. Charouh, M. Ghogho, and Z. Guennoun, "Robust deep learning-based driver distraction detection and classification," *IEEE Access*, vol. 9, pp. 168 080–168 092, 2021.

[20] A. Kashevnik, R. Shchedrin, C. Kaiser, and A. Stocker, "Driver distraction detection methods: A literature review and framework," *IEEE Access*, vol. 9, pp. 60 063–60 076, 2021.

[21] B. Roytburd, A. Shaout, and L. A. Sanchez-Perez, "An embedded deep learning computer vision method for driver distraction detection," *Available at SSRN 3996984*, 2021.

[22] F. Sajid, A. R. Javed, A. Basharat, N. Kryvinska, A. Afzal, and M. Rizwan, "An efficient deep learning framework for distracted driver detection," *IEEE Access*, vol. 9, pp. 169 270–169 280, 2021.

[23] N. C. for Statistics and Analysis, "Distracted driving 2019," *National Highway Traffic Safety Administration*, Apr. 2021, (Report No. DOT HS 813 111). [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813111.

[24] N. C. for Statistics and Analysis, "Distracted driving 2020," *National Highway Traffic Safety Administration*, Apr. 2021, (Report No. DOT HS 813 184). [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813184.

[25] M. H. Alkinani, W. Z. Khan, Q. Arshad, and M. Raza, "Hsddd: A hybrid scheme for the detection of distracted driving through fusion of deep learning and handcrafted features," *Sensors*, vol. 22, no. 5, p. 1864, 2022.

[26] T. Covington, "Distracted driving statistics: Research and facts in 2022," *The Zebra*, May 2022. [Online]. Available: https://www.thezebra.com/resources/research/distracted-driving-statistics/.

[27] X. Fu, H. Meng, X. Wang, H. Yang, and J. Wang, "A hybrid neural network for driving behavior risk prediction based on distracted driving behavior data," *PloS one*, vol. 17, no. 1, e0263030, 2022.

[28] Y. Li, L. Wang, W. Mi, H. Xu, J. Hu, and H. Li, “Distracted driving detection by combining vit and cnn,” in *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, IEEE, 2022, pp. 908–913.