Benha University

Benha Faculty of Engineering

Electrical Engineering Department

DRIVER DISTRACRION DETECTION

BY

Mohamed Osama Anwer

Mamdouh Mohammed El-maghwry

Nahla Hatem Mohamed

Huda Ahmed Mohamed

Haidy Ashraf Eid Saad

Heba Mohamed Abd-elmonam

Yasmin Mohamed Abd-elstar

Under the supervision of

DR. Radwa Tawfeek

Benha University

July 2021

ACKNOWLEDGMENT

It has been a great opportunity to gain lots of experience in real-time projects, followed by the knowledge of how to design and analyze real projects. For that, we want to thank all the people who made it possible for students like us. Special thanks to the graduation Project Unit for the efforts they did to provide us with all useful information and making the path clear for the students to implement all the education periods in real-time project design and analysis. Furthermore, we all the professors and visiting industry for the interesting lectures they presented which had great benefit for all of us. We would like to express our deepest gratitude to Dr. Radwa Tawfeek, our project supervisor, for her patient guidance, enthusiastic encouragement, and useful critiques of this research work. We would like to express our gratitude towards our parents and members of BENHA FACULTY OF ENGINEER-ING for their kind co-operation and encouragement which helped us in finishing this project.

ABSTRACT

Driver distraction, defined as the diversion of attention away from activities critical for safe driving toward a competing activity, is increasingly recognized as a significant source of injuries and fatalities on the roadway. Additionally, the trend towards increasing the use of in-vehicle information systems is critical because they induce visual, biomechanical, and cognitive distraction and may affect driving performance in qualitatively different ways. Non-intrusive methods are strongly preferred for monitoring distraction, and vision-based systems have appeared to be attractive for both drivers and researchers.

we attempt to develop an accurate and robust system for detecting distracted driver actions and alert him against them. Motivated by the performance of Convolutional Neural Networks in computer vision, we present a CNN-based system that not only detects the distracted driver but also identifies the cause of distraction. VGG-16 architecture is modified for this particular task and various regularization techniques are implied to improve the performance. and also build a CNN model from scratch. Experimental results show that our system outperforms earlier methods in literature achieving a validation accuracy of 99.5% and processes 5 images per second on Raspberry pi GPU. We also study the effect of dropout, augmentation, and batch normalization on the performance of the system. As well as the effect of images size, the number of layers, and the optimizer on the training time.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

The distracted driving is any activity that diverts attention from driving, including talking or texting on your phone, eating and drinking, talking to people in your vehicle, fiddling with the stereo, entertainment or navigation system — anything that takes your attention away from the task of safe driving. You cannot drive safely unless the task of driving has your full attention. Any non-driving activity you engage in is a potential distraction and increases your risk of crashing.

1.1 The Problem Of Distracted Driving

According to the World Health Organization (WHO) survey, 1.3 million people worldwide die in traffic accidents each year, making them the eighth leading cause of death and an additional 20-50millions are injured/ disabled. As per the report of National Crime Research Bureau (NCRB), Govt. of India, Indian roads account for the highest fatalities in the world. There has been a continuous increase in road crash deaths in India since 2006. Also, the total number of deaths have risen to 1.46 lakhs in 2015 and driver error is the most common cause behind these traffic accidents. The number of accidents because of distracted driver has been increasing since few years.

National Highway Traffic Safety Administrator of United States (NHTSA) reports deaths of 3477 people and injuries to 391000 people in motor vehicle crashes because of distracted drivers in 2015. In the United States, everyday approximately 9 people are killed and more than 1,000 are injured in road crashes that are reported to involve a distracted driver. NTHSA describes distracted driving as *"any activity that diverts attention of the driver from the task of driving"* which can be classified into Manual, Visual or Cognitive distraction. As per the definitions of Center for Disease Control and Prevention (CDC), cognitive distraction is basically "driver's mind is off the driving".

1.2 The Three Types of Distracted Driving

Even though the driver is in safe driving posture, he is mentally distracted from the task of driving. He might be lost in thoughts, daydreaming etc. Distraction because of inattention, sleepiness, fatigue or drowsiness falls into visual distraction class where "drivers' eyes are off the road". Manual distractions are concerned with various activities where "driver's hands are off the wheel". Such distractions include talking or texting using mobile phones, eating and drinking, talking to passengers in the vehicle, adjusting the radio, makeup etc.

1.3 Project Overview

In this project, we focus on detecting manual distractions where driver is engaged in other activities than safe driving to prevent accidents by offering technologies that alert the driver to potential problems and to keep the car's driver and occupants safe if an accident does occur. We present a Convolutional Neural Network based approach for this problem. We also attempt to reduce the computational complexity and memory requirement while maintaining good accuracy which is desirable in real time applications.

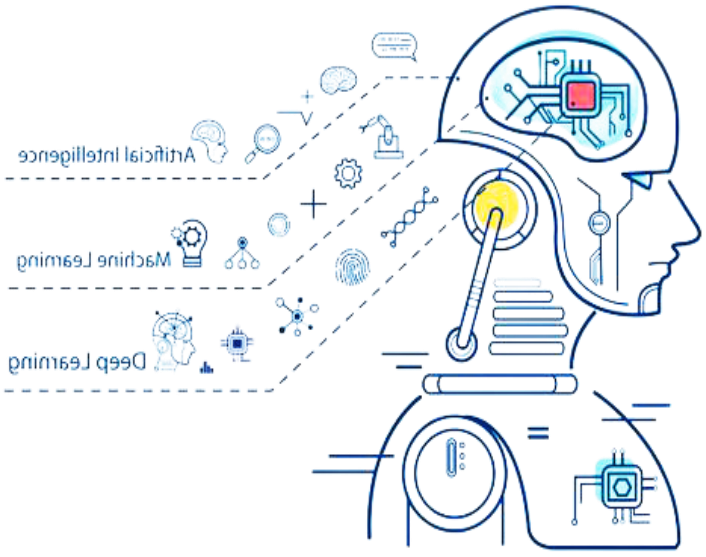
A deep-learning approach is used, which is one of the artificial intelligence-based machine-learning methodologies. Computer vision, speech recognition, natural language processing, and audio recognition are just a few of the disciplines where deep learning is used. In this work, experimental work was conducted on driver images to detect if the drivers are distracted while driving.

As discussed above, the consequences of distracted driving can be grave. Driver distraction is classified into ten different classes based on driver actions while driving. In this project, the StateFarm dataset is used, and the convolution-neural-network (CNN) technique is applied to learn the machine and further classify the real image.

The rest of the book is structured as follows. The background is covered in Chapter 2, while the technical procedure, experimental results and model evaluations are covered in Chapter 3. Chapter 4 covered the connection of hardware components, the development of a suitable environment, and how the device uses the Pi camera to capture driver activities. Chapter 5 includes a database description connected to the detection model, as well as a discussion of the Graphical User Interface (GUI). Finally, conclusions and future work are presented in Chapter 6.

BACKGROUND

Less than a decade after breaking the Nazi encryption machine Enigma and helping the Allied Forces win World War II, mathematician Alan Turing changed history a second time with a simple question *"Can machines think?"*. Turing's paper"Computing Machinery and Intelligence" (1950), and its subsequent Turing Test, established the fundamental goal and vision of artificial intelligence. At its core, AI is the branch of computer science that aims to answer Turing's question in the affirmative. It is the endeavor to replicate or simulate human intelligence in machines. The expansive goal of artificial intelligence has given rise to many questions and debates. So much so, that no singular definition of the field is universally accepted. The major limitation in defining AI as simply *"building machines that are intelligent"* is that it doesn't actually explain what artificial intelligence is? What makes a machine intelligent in their groundbreaking textbook Artificial Intelligence: A Modern Approach, authors (*Stuart Russell)* and (*Peter Norvig)* approach the question by unifying their work around the theme of intelligent agents in machines. AI is *"the study of agents that receive percepts from the environment and perform actions." (Russel and Norvig).*



2.1 Artificial intelligence

Norvig and Russell go on to explore four different approaches that have historically defined the field of AI.

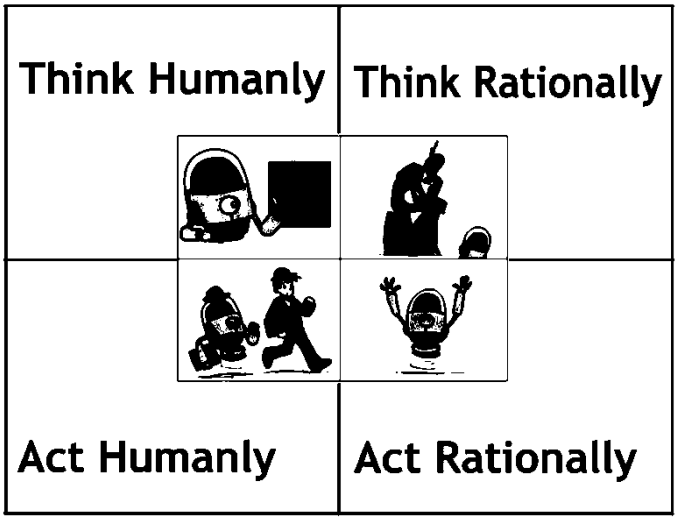


Figure 1: Four different approaches of AI.

The first two ideas concern thought processes and reasoning, while the others deal with behavior. Norvig and Russell focus particularly on rational agents that act to achieve the best outcome, noting *"all the skills needed for the Turing Test also allow an agent to act rationally."*

(Patrick Winston), the Ford professor of artificial intelligence and computer science at MIT, defines AI as *"Algorithms enabled by constraints, exposed by representations that support models targeted at loops that tie thinking, perception and action together."*

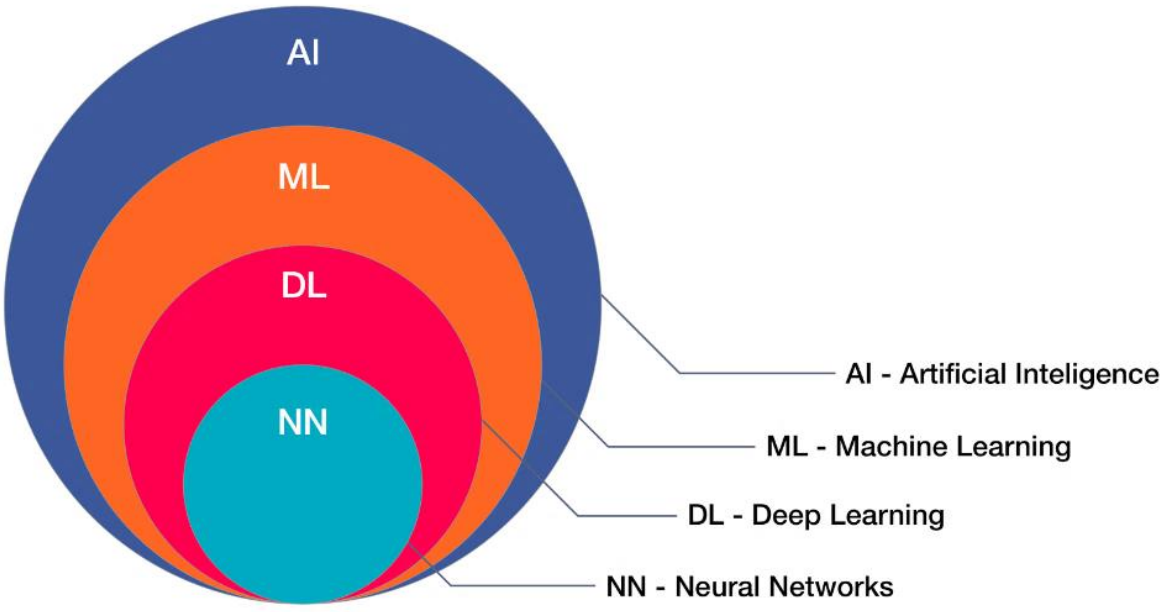


Figure 2: Artificial intelligence development and expansion.

2.2 Machine Learning

When most people hear *"Machine Learning"* they picture a robot: a dependable butter or a deadly Terminator depending on who you ask. But Machine Learning is not just a futuristic fantasy, it's already here. In fact, it has been around for decades in some specialized applications, such as Optical Character Recognition (OCR). But the first ML application that really became mainstream, improving the lives of hundreds of millions of people, took over the world back in the 1990s: it was the spam filter. Not exactly a self-aware Skynet, but it does technically qualify as Machine Learning (it has actually learned so well that you seldom need to flag an email as spam any- more). It was followed by hundreds of ML applications that now quietly power hundreds of products and features that you use regularly, from better recommendations to voice search.

Have you ever wondered how you send someone a text about something you want to buy and suddenly out of the blue you find and advertisement about that thing on Facebook, Google, or whatever site you are on, or when you put a photo of you and a friend on Facebook and you find that Facebook recognized the face of your friend and maybe tagging him automatically? It is not someone spying on you as much as it is a machine learning algorithm spying on you training on the data you share and use to advertise or help you in many things.

2.2.1 The Advantages Of Using Machine Learning

Resurging interest in machine learning is due to the same factors that have made data mining and Bayesian analysis more popular than ever. Things like growing volumes and varieties of available data, computational processing that is cheaper and more powerful, and affordable data storage.

- Problems for which existing solutions require a lot of hand-tuning or long lists of rules: one Machine Learning algorithm can often simplify code and perform better.
- Complex problems for which there is no good solution at all using a traditional approach: the best Machine Learning techniques can find a solution.
- Fluctuating environments: A machine Learning system can adapt to new data.

All of these things mean it's possible to quickly and automatically produce models that can analyze bigger, more complex data and deliver faster, more accurate results even on a very large scale. And by building precise models, an organization has a better chance of identifying profitable opportunities or avoiding unknown risks.

2.2.2 Types Of Machine Learning Algorithm

Supervised Learning Algorithms

- Binary Classification - Algorithm that can be used for Binary Classification includes:
  - Decision Trees
  - k-Nearest Neighbors
  - Logistic Regression
  - Naive Bayes
  - Support Vector Machine
- Multi-class Classification - Many algorithms that used in binary classification can also be used for multi-class classification. Algorithms that can be used for Multi-class Classification includes:
  - Decision Trees
  - Gradient Boosting
  - k-Nearest Neighbors
  - Naive Bayes
  - Random Forest
- Regression - is the task of predicting a continuous quantity that requires the prediction of a quantity. A regression problem with multiple input variables is called a multivariate regression problem. An example of a regression application is predicting the price of a stock over a period of time.

Unsupervised Learning Algorithms

In unsupervised learning, the machine is trained on un-labelled data without any guidance. No idea which types of results are expected. it can be thought of as self-learning where the algorithm can find previously unknown patterns in datasets that does not have any sort of labels. Used in recommender system, fake news identification, etc. are some examples of unsupervised learning.

The two types of Unsupervised Learning are:

- Association – Association analysis is the task of uncovering relationship among data i.e., discovering patterns in data, finding co-occurrences and so on. A classic example of association rule mining is the relationship between bread and jam. So, people who tend to buy bread also tend to buy jam. Overall, it is all about finding associations between items that frequently co-occur or items that are similar to each other.

- Clustering - the process of finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters. A good clustering method will produce high quality clusters with:
  - High intra-class similarity
  - Low inter-class similarity
- Example: Digital AdWords use a clustering technique to cluster potential buyers into different categories based on their interests and intents.
- Different types of Clustering Algorithms are:
  - Hierarchical Method
  - Non-hierarchical Method

Reinforcement learning

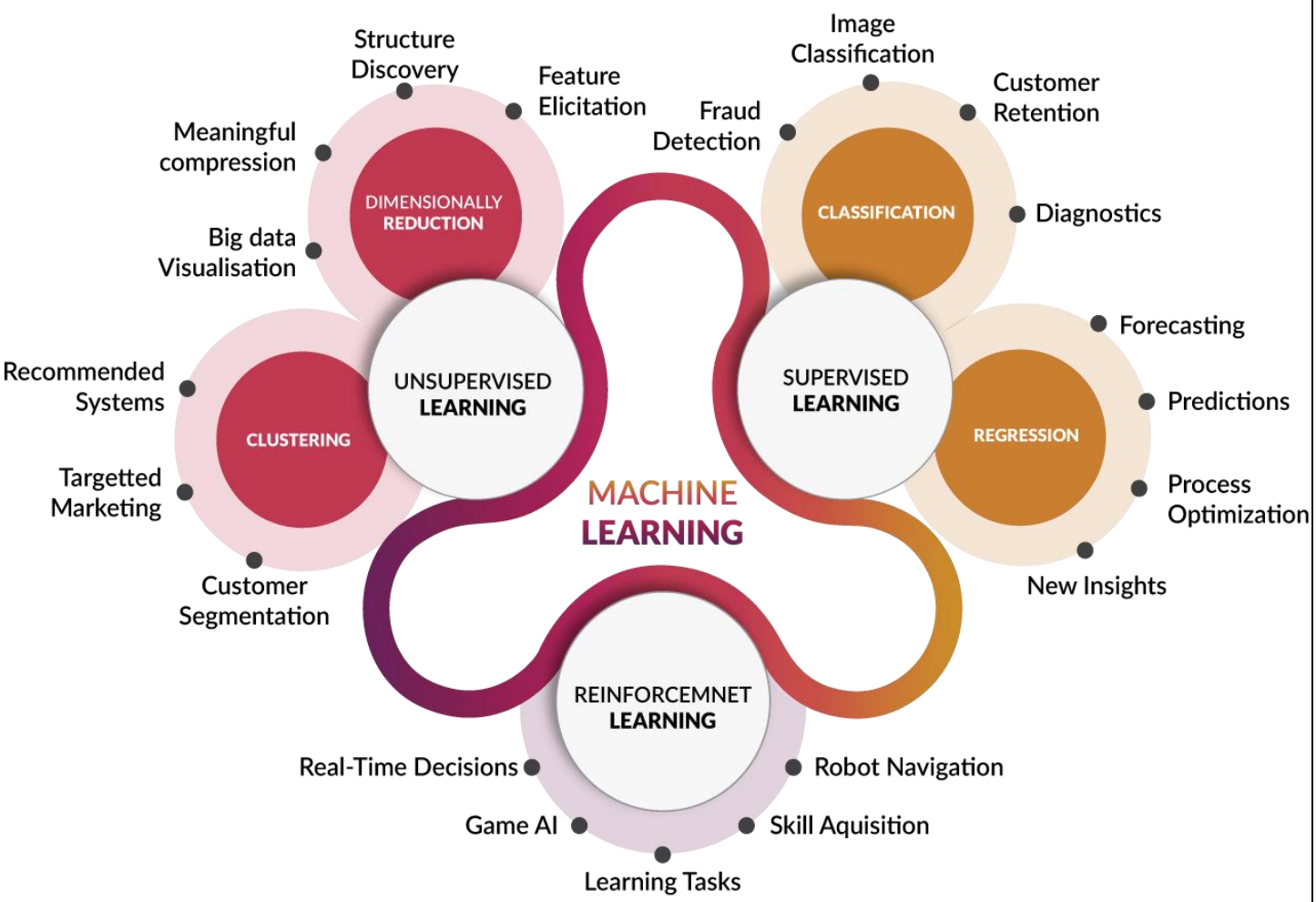


Figure 3: Types of machine learning systems.

2.3 Deep Learning

In 2006, Geoffrey Hinton et al. published a paper1 showing how to train a deep neural network capable of recognizing handwritten digits with state-of-the-art precision (>98%). They branded this technique *"Deep Learning."* Training a deep neural net was widely considered impossible at the time, the two and most researchers had abandoned the idea since the 1990s. This paper revived the interest of the scientific community and before long many new papers demonstrated that Deep Learning was not only possible, but capable of mind-blowing achievements that no other Machine Learning (ML) technique could hope to match (with the help of tremendous computing power and great amounts of data). This enthusiasm soon extended to many other areas of Machine Learning.

Fast-forward 10 years and Machine Learning has conquered the industry: it is now at the heart of much of the magic in today's high-tech products, ranking your web search results, powering your smartphone's speech recognition, and recommending videos, beating the world champion at the game of Go. Before you know it, it will be driving your car.

ANNs are at the very core of Deep Learning. They are versatile, powerful, and scalable, making them ideal to tackle large and highly complex Machine Learning tasks, such as classifying billions of images (e.g., Google Images), powering speech recognition services (e.g., Apple's Siri), recommending the best videos to watch to hundreds of millions of users every day (e.g., YouTube), or learning to beat the world champion at the game of Go by examining millions of past games and then playing against itself (DeepMind's AlphaGo).

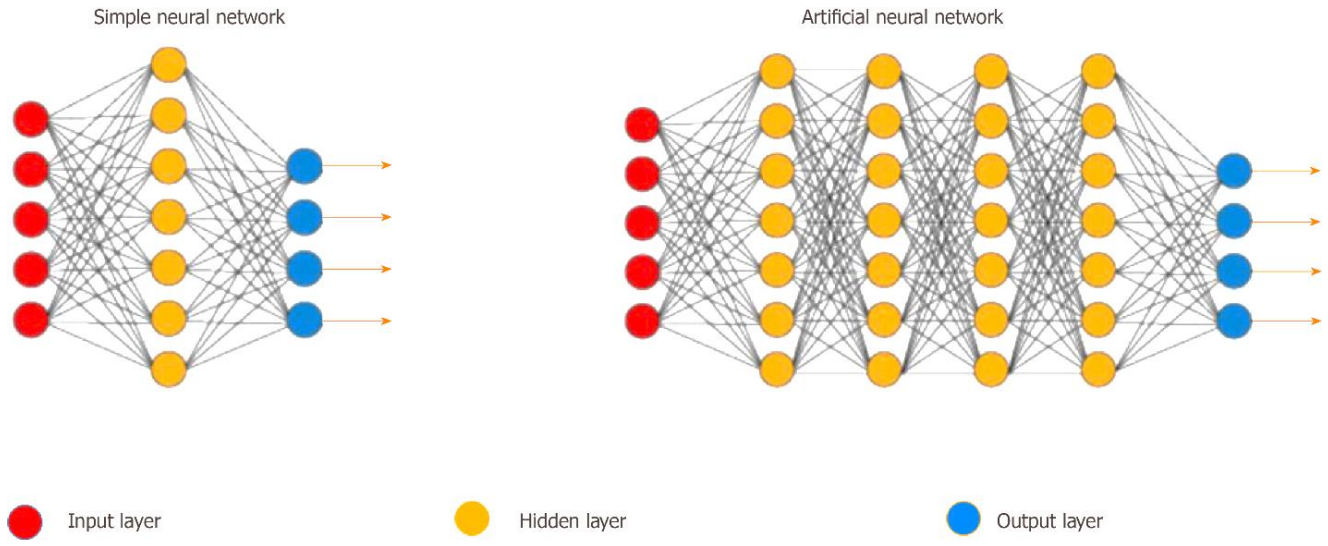


Figure 4: Deep Learning layers.

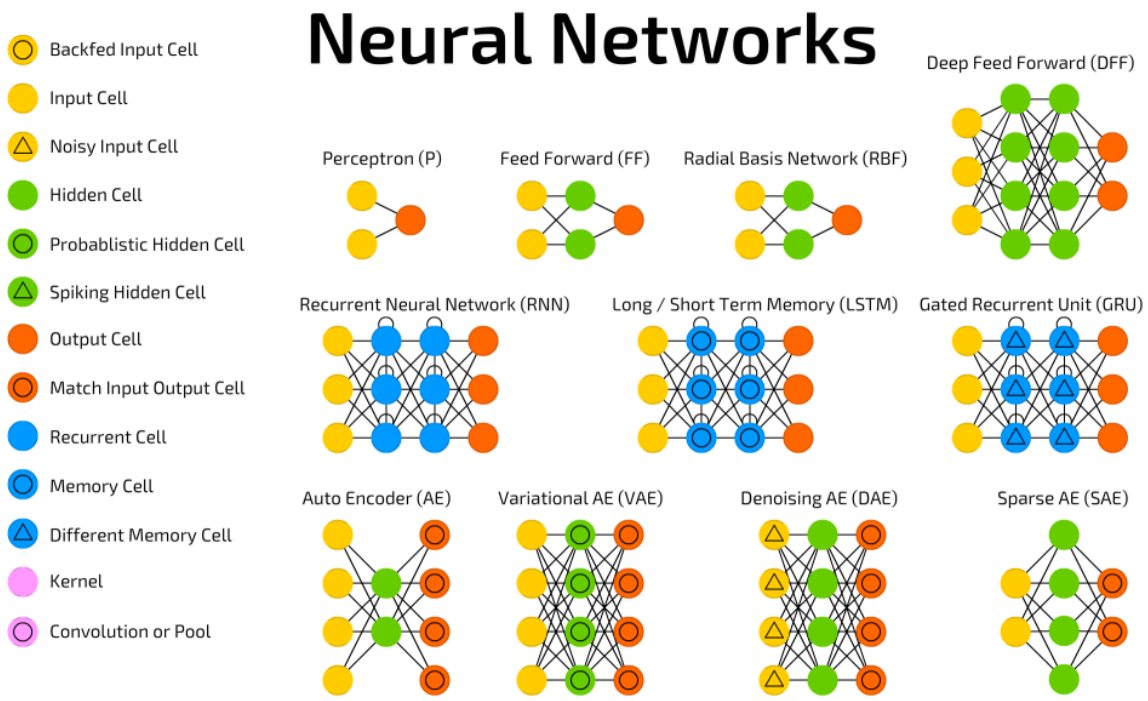


Figure 5: A mostly complete chart of Artificial Neural Network.

2.3.1 Convolutional Neural Networks

Although IBM's Deep Blue supercomputer beat the chess world champion Garry Kasparov back in 1996, until quite recently computers were unable to reliably perform seemingly trivial tasks such as detecting a puppy in a picture or recognizing spoken words. Why are these tasks so effortless to us humans? The answer lies in the fact that perception largely takes place outside the realm of our consciousness, within specialized visual, auditory, and other sensory modules in our brains. By the time sensory information reaches our consciousness, it is already adorned with high-level features; for example, when you look at a picture of a cute puppy, you cannot choose *not* to see the puppy, or *not* to notice its cuteness. Nor can you explain how you recognize a cute puppy; it's just obvious to you. Thus, we cannot trust our subjective experience: perception is not trivial at all, and to understand it we must look at how the sensory modules work.

Convolutional neural networks (CNNs) emerged from the study of the brain's visual cortex, and they have been used in image recognition since the 1980s. In the last few years, thanks to the increase in computational power, the amount of available training data, CNNs have managed to achieve superhuman performance on some complex visual tasks. The power image search services, self-driving cars, automatic video classification systems, and more. Moreover, CNNs are not restricted to visual perception: they are also

successful at other tasks, such as voice recognition or Natural Language Processing (NLP); however, we will focus on visual applications for now.

A CNN sequence to classify handwritten digits: It takes in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

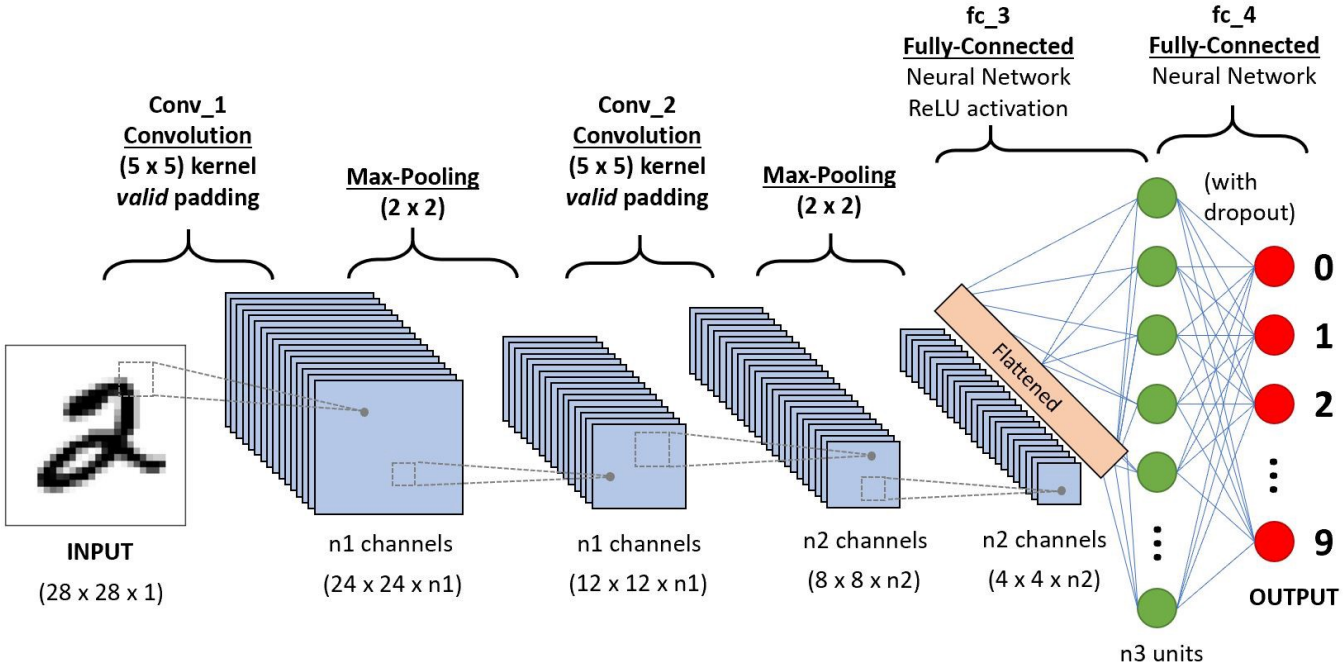


Figure 6: CNN architecture for classification application.

The pre-processing required in a convolutional neural network (ConvNets) is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNets is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimulionly in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

ConvNets' advantages over Feed-Forward Neural Networks

Flattening of a 3x3 image matrix into a 9x1 vector .An image is nothing but a matrix of pixel values, right? So why not just flatten the image (e.g., 3x3 image matrix into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification purposes.

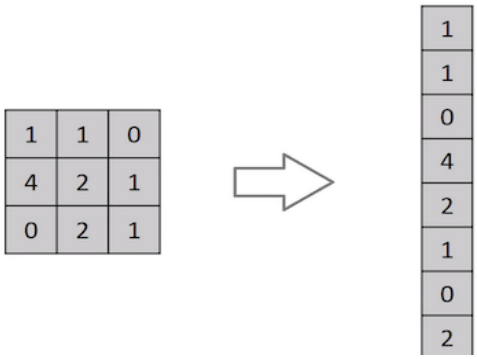


Figure 7: Flatten a 3x3 image.

In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout.

A ConvNets can successfully capture the spatial and temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

RGB Input Image

In the figure, we have an RGB image which has been separated by its three-color planes — Red, Green, and Blue. There are several color spaces in which images exist — Grayscale, RGB, HSV, CMYK, etc.

You can imagine how computationally intensive things would get once the images reach 4x4x3 RGB Image dimensions, say 8K (7680×4320). The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

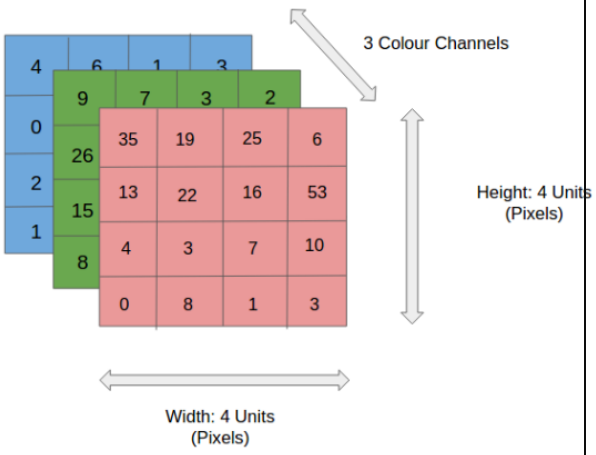


Figure 8: A RGB image

The Kernel In Convolution Neural Network

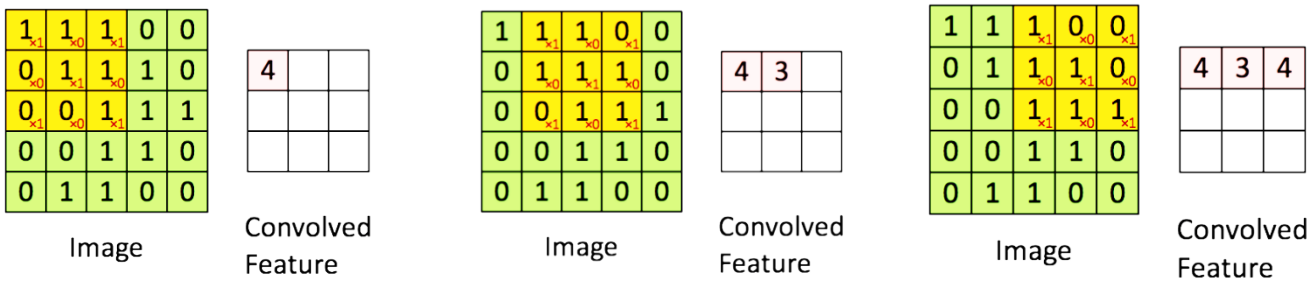


Figure 9: Convoluting a 5x5x1 image with a 3x3x1 kernel.

Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature. The filter moves to the right with a certain stride value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

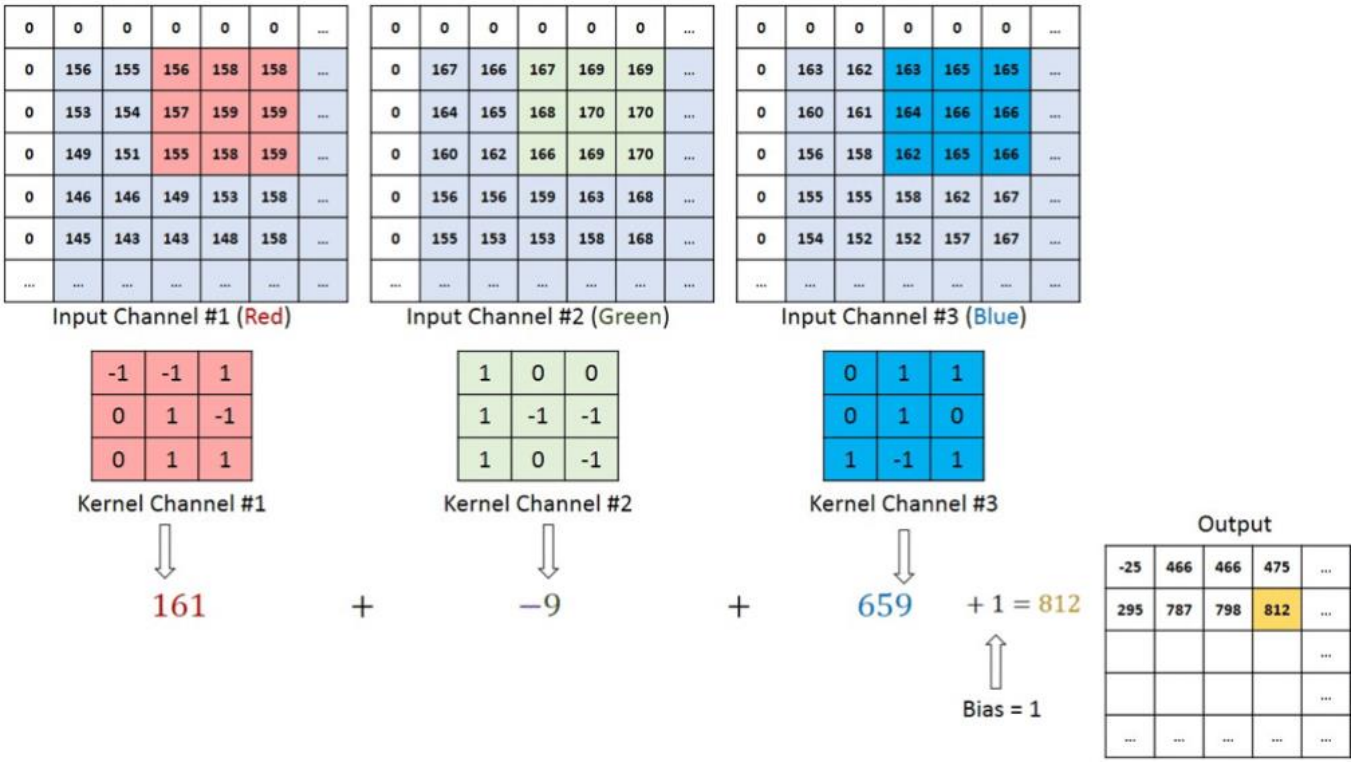


Figure 11: Convoluting a RGB image with a three channels kernel.

In the case of images with multiple channels (e.g., RGB), the Kernel has the same depth as that of the input image. Matrix multiplication is performed between Kn and in stack ([K1, I1], [K2, I2], [K3, I3]) and all the results are summed with the bias to give us a squashed one-depth channel convoluted feature output.

Convolution operation on a MxNx3 image matrix with a 3x3x3 Kernel

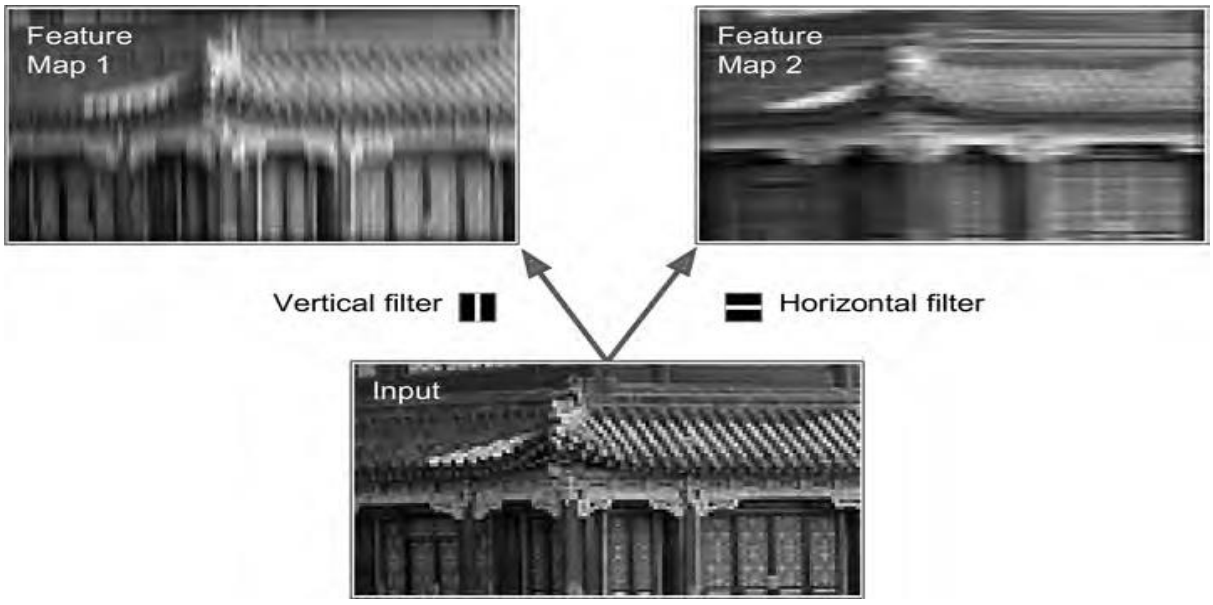


Figure 10: Vertical filter Vs Horizontal filter.

The objective of the convolution operation is to extract the high-level features such as edges, from the input image. ConvNets need not be limited to only one convolutional layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network, which has the wholesome understanding of images in the dataset, like how we would.

Now if all neurons in a layer use the same vertical line filter (and the same bias term), and you feed the network the input image shown in Figure 10, the layer will output the top-left image. Notice that the vertical white lines get enhanced while the rest gets blurred. Similarly, the upper-right image is what you get if all neurons use the horizontal line filter. Notice that the horizontal white lines get enhanced while the rest is blurred out. Thus, a layer full of neurons using the same filter gives.

A feature map, which highlights the areas in an image that are most similar to the filter. During training, a CNN finds the most useful filters for its task, and it learns to combine them into more complex patterns (e.g., a cross is an area in an image where both the vertical filter and the horizontal filter are active).

Padding and Stride

Before going deeper into convolution, you need to know one extra thing, actually there are two important concepts you need to know and fully grasp. Both padding and stride can change the way your model sees the input observation. Also, changing their parameters impacts the shape of the output feature map.

Although the convolutional layer is very simple, it is capable of achieving sophisticated and impressive results. Nevertheless, it can be challenging to develop an understanding for how the shape of the filters impacts the shape of the output feature map and how related configuration hyperparameters such as padding, and stride should be configured.

In a convolution operation we have a kernel, and to make the final output of the operation more informative we use padding in an image matrix or any kind of input array. Adding padding to the input makes the kernel start giving more information to the edges of the input observation, thus making all the information and features hidden in edges of the input appear in our output.

There are three types of padding, stated as follows:

- **Padding full:** This type shows the importance of extracting the information from the edges of the input. When you use full padding on the input, it makes the kernel in the convolution operation treat each pixel with the same priority, which means the kernel steps over the edges with the same amount as the center pixels, and this may work for increasing the dimensional of an input observation (image).

- **Padding same**: In this type of padding, we need to make the output observation shape from convolution operation get the same shape as the input observation. For instance, if we have a 32×32 image as input, the output will have the same shape, 32×32.
- **Padding valid**: Simply, valid convolution means no padding at all, and this may work for you as a dimensionality reduction for an input observation (image). For instance, an image with 32×32 input with kernel filter of 3×3 will generate a 30×30 output image.

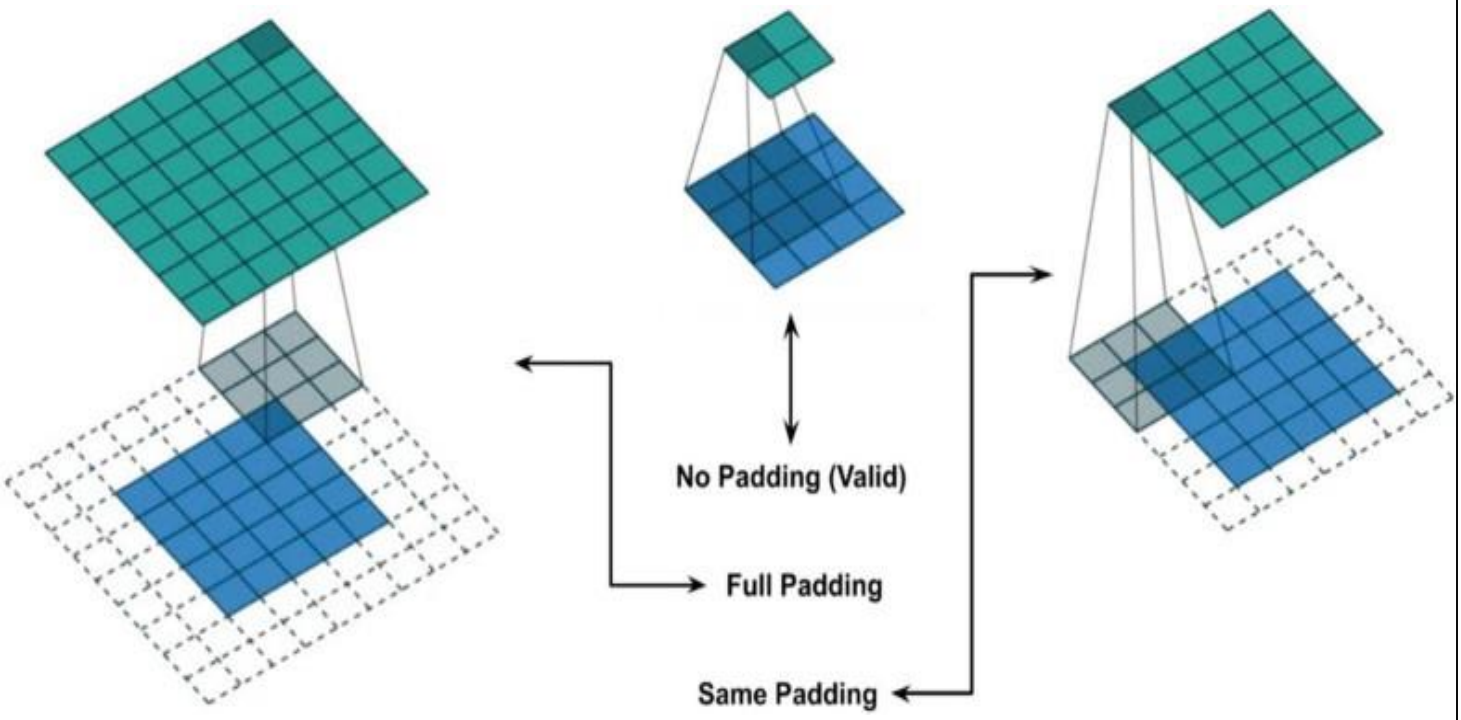


Figure 12: The three types of paddings.

Pooling Layers

Convolutional layers in a convolutional neural network systematically apply learned filters to input images in order to create feature maps that summarize the presence of those features in the input. Convolutional layers prove very effective and stacking convolutional layers in deep models allows layers close to the input to learn low-level features (e.g., lines) and layers deeper in the model to learn high-order or more abstract features, like shapes or specific objects. A limitation of the feature map output of convolutional layers is that they record the precise position of features in the input. This means that small movements in the position of the feature in the input image will result in a different feature map. This can happen with re-cropping, rotation, shifting, and other minor changes to the input image.

A common approach to addressing this problem from signal processing is called down sampling. This is where a lower resolution version of an input signal is created that still contains the large or important structural elements, without the fine detail that may not be as useful to the task. This down sampling can be achieved with convolutional layers by changing the stride of the convolution across the image.

A more robust and common approach is to use a pooling layer. The pooling layer is a new layer added after the convolutional layer. Specifically, after a nonlinearity (e.g., ReLU) has been applied to the feature maps output by a convolutional layer.

Input Image
Convolutional Layer
Non-linearity (AF)
Pooling Layer

The addition of a pooling layer after the convolutional layer is a common pattern used for ordering layers within a convolutional neural network that may be repeated one or more times in a given model. The pooling layer operates upon each feature map separately to create a new set of the same number of pooled feature maps. Pooling involves selecting a pooling operation, much like a filter to be applied to feature maps. The size of the pooling operation or filter is smaller than the size of the feature map. Specifically, it is almost always 2×2 pixels applied with a stride of 2 pixels.

This means that the pooling layer will always reduce the size of each feature map by a factor of 2, (e.g., each dimension is halved), reducing the number of pixels or values in each feature map to one quarter the size. For example, a pooling layer applied to a feature map of 6×6 (36 pixels) will result in an output pooled feature map of 3×3 (9 pixels).

The pooling operation is specified, rather than learned. Two common functions used in the pooling operation are:

- **Average Pooling -** Calculate the average value for each patch on the feature map.
- **Maximum Pooling -** Calculate the maximum value for each patch of the feature map.

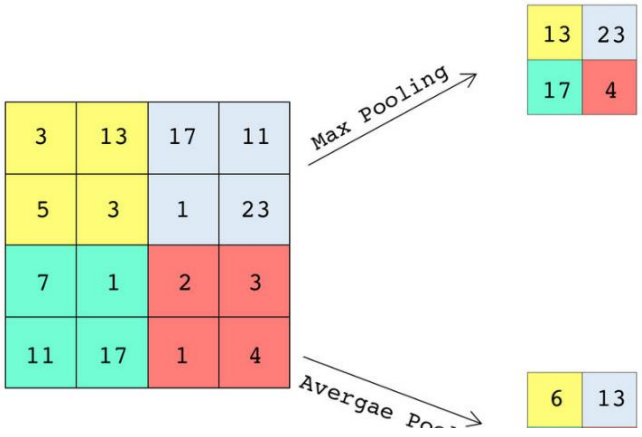


Figure 13: Average pooling VS Max pooling

Max Pooling also performs as a noise suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling.

The result of using a pooling layer and creating down sampled or pooled feature maps is a summarized version of the features detected in the input. They are useful as small changes in the location of the feature in the input detected by the convolutional layer will result in a pooled feature map with the feature in the same location. This capability added by pooling is called the model's invariance to local translation.

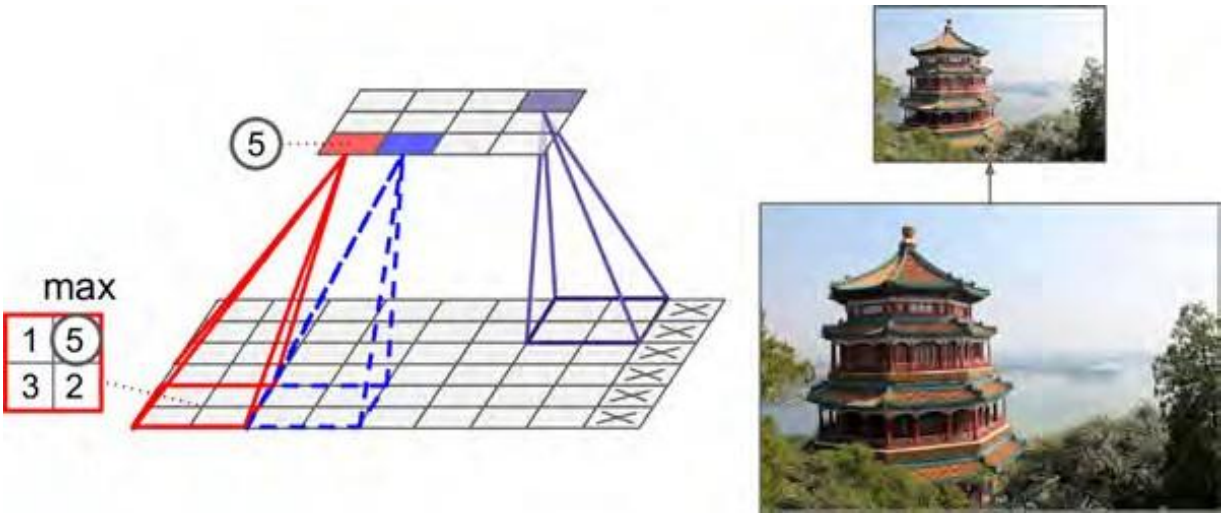


Figure 14: Max pooling layer (2 × 2 pooling kernel, stride 2, no padding).

Fully Connected Layer (FC Layer)

Adding a Fully Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully Connected layer is learning a possibly non-linear function in that space.

Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the SoftMax Classification technique.

CNN Architectures

Building the CNN architecture

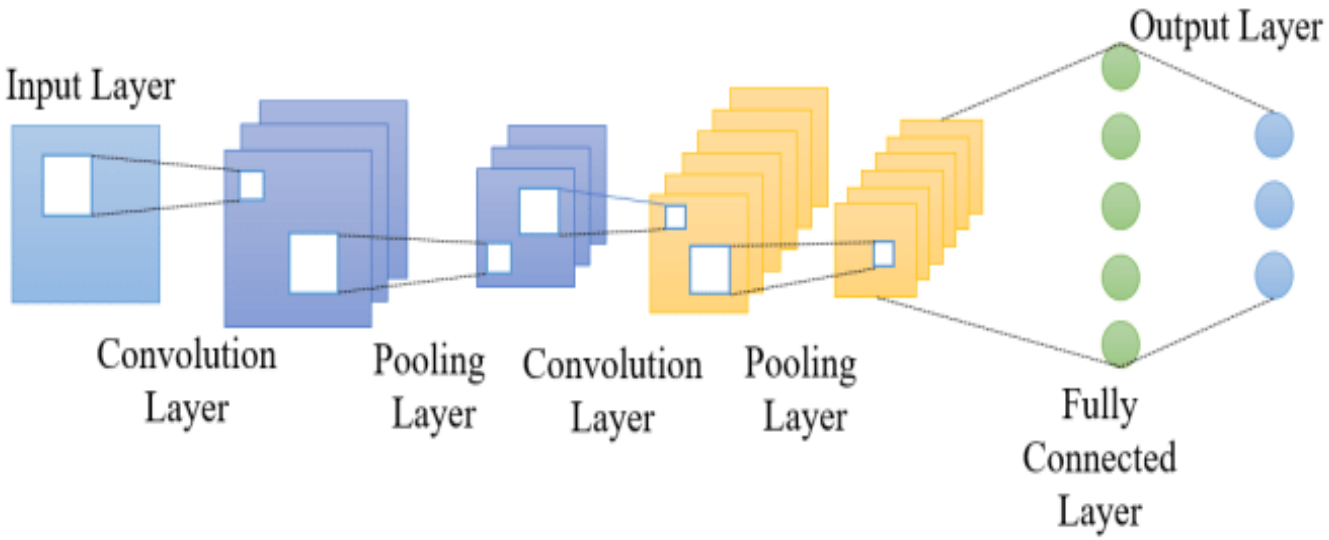


Figure 15: CNN model architecture.

we need to design CNN architectures and then fine-tune hyper-parameters to get our best model. we give a set of values for all the hyperparameters and let the module decide which is best won't work for tuning CNN. This is because at each layer the input dimensions decrease due to operations like convolution and max-pooling hence if we give a range of values for hyperparameters like stride, filter-size etc.

Step-1: make a sheet containing the dimensions, activation shapes and sizes for the architecture just as shown below.

Table 1: Example of a model architecture sheet.

Layers	Number	padding	Output	Parameters
Input image	-	-	(28, 28, 1)	784
Conv2D_1	8	Same	(28,28,8)	6,272
MaxPool-	-	Valid	(14, 14, 8)	1,568
Conv2D_2	16	Valid	(10, 10, 16)	1,600
MaxPool-	-	valid	(5, 5, 16)	400
Flatten	-	-	(400, 1)	400
Dense	-	-	(120, 1)	120
Dense	-	-	(64,1)	64
Softmax	-	-	(10,1)	10

Step-2: we calculate the Activation shape after every convolution or pooling operation is [ceil(N+f-1)/s, ceil(N+f-1)/s, Number of filters]. wherever the padding value is 'valid' and dimensions are [N, N, Number of filters] where the padding used is 'same', here 'N' is input dimensions 'f' is filter size and 's' is stride length. The Activation value is computed by multiplying all values in the dimensions of activation shape.

Con2D_1: N=28, f=3, s=1, Number of filters=8 and padding is 'same' hence we get Activation shape as (28,28,8). Activation Value= 28x28x8=6272.

Max-Pool_1: N=28, f=2, s=2 and padding is 'valid' hence we get Activation shape as ((28+2-1)/2, (28-2+1)/2, 8) = (14,14,8). Activation Value= 14x14x8=1568.

Con2D_2: N=14, f=5, s=1, Number of filters = 16 and padding is 'valid' hence we get Activation shape as ((14-5+1)/1, (14-5+1)/1, 16) = (10,10,16). Activation Value= 10x10x16=1600.

Max-Pool_2: N=10, f=2, s=2 and padding is 'valid' hence we get Activation shape as ((10+2-1)/2, (10+2-1)/2, 8) = (5,5,16). Activation Value= 5x5x16=400.

Flatten: makes the input into a one-dimensional list for input to the dense layers therefore Activation shape is (400,1). Activation Value= 400x1=400.

Dense_1: Activation shape (120, 1) states 120 hidden units are used in the dense layer. Activation Value = 120x1 = 120.

Dense_2: Activation shape (64, 1) states 64 hidden units are used in the dense layer. Activation Value = 64x1 = 64.

This computation tells us if we are choosing the right parameters while building our CNN architecture as the architecture must not end up with negative dimensions by going overboard with usage of high values of stride length and filter-sizes. It is very important to ensure that you choose your values such that there is a general decreasing trend in the Activation values and there aren't any very abrupt changes in Activation values.

Pre-trained Models Architecture

There are various architectures of CNNs available which have been key in building algorithms which power and shall power AI as a whole in the foreseeable future. Some of them have been listed below:

- AlexNet.
- VGGNet.
- GoogLeNet.
- ResNet.

2.3.2 Types of Activation Functions

Rectified Linear Unit (ReLU) Function

One of the most popular activation functions in deep learning models, the rectified linear unit (ReLU) function, is a fast-learning activation functions that promises to deliver state-of-the-art performance with stellar results. Compared to other activation functions like the sigmoid and tanh functions, the ReLU function offers much better performance and generalization in deep learning. The function is a nearly linear function that retains the properties of linear models, which makes them easy to optimize with gradient-descent methods. Specifically, it is less susceptible to vanishing gradients that prevent deep models from being trained, although it can suffer from other problems like saturated or "dead" units.

The ReLU function performs a threshold operation on each input element where all values less than zero are set to zero. Thus, the ReLU is represented as:

$$f(x) = \max(0, x) = \begin{cases} x, & if \quad x \geq 0 \\ 0, & if \quad x < 0 \end{cases}$$

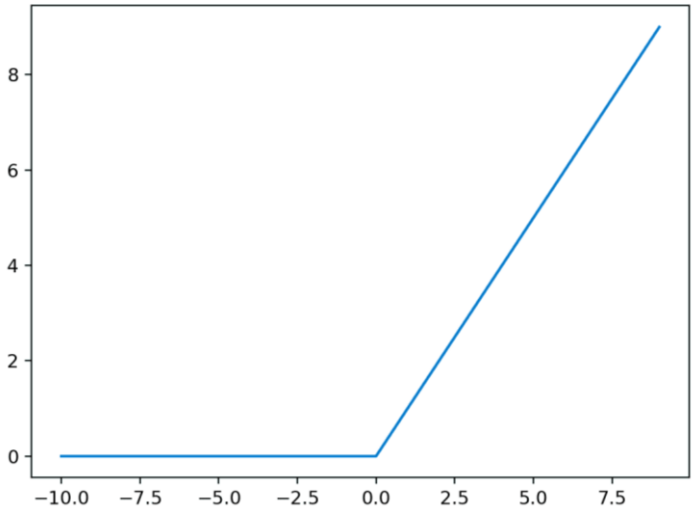


Figure 16: Rectified Linear Unit (ReLU) Function.

This means that if the input value (x) is negative, then a value 0.0 is returned, otherwise, the value is returned. By rectifying the values of the inputs less than zero and setting them to zero, this function eliminates the vanishing gradient problem observed in the earlier types of activation functions (sigmoid and tanh). The most significant advantage of using the ReLU function in computation is that it guarantees faster computation – it does not compute exponentials and divisions, thereby boosting the overall computation speed. Another critical aspect of the ReLU function is that it introduces sparsity in the hidden units by squishing the values between zero to maximum.

Sigmoid Function

In an ANN, the sigmoid function is a non-linear activation function used primarily in feedforward neural networks. It is a differentiable real function, defined for real input values, and containing positive derivatives everywhere with a specific degree of smoothness. The sigmoid function appears in the output layer of the deep learning models and is used for predicting probability-based outputs. The sigmoid activation function is also called the logistic function. It is the same function used in the logistic regression classification algorithm.

The function takes any real value as input and outputs values in the range 0 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to 0.0. The sigmoid function is represented as:

$$\sigma(x) = \left(\frac{1}{1 + e^{-x}}\right)$$

$$\tilde{\sigma}(x) = \sigma(x)(1 - \sigma(x))$$

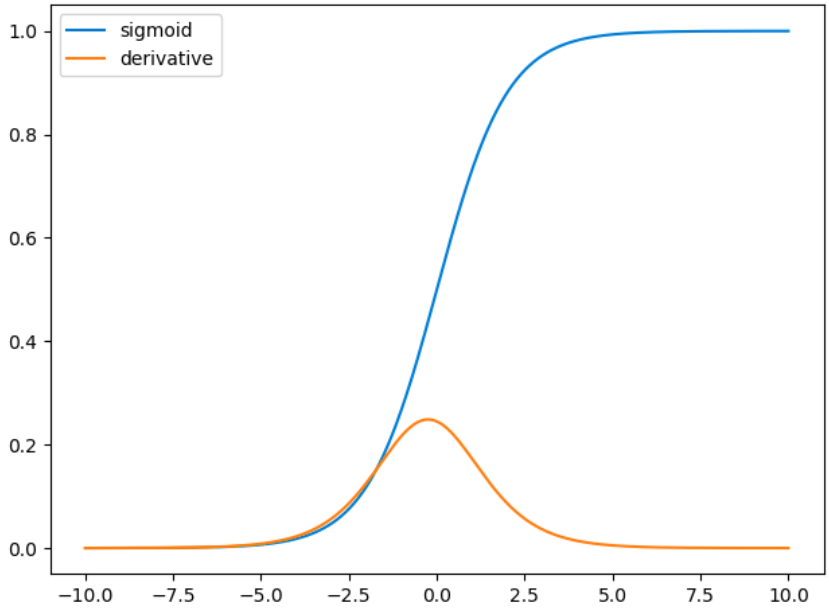


Figure 17: Sigmoid Activation function and its derivative.

Generally, the derivatives of the sigmoid function are applied to learning algorithms. The graph of the sigmoid function is 'S' shaped. Some of the major drawbacks of the sigmoid function include gradient saturation, slow convergence, sharp damp gradients during backpropagation from within deeper hidden layers to the input layers, and non-zero centered output that causes the gradient updates to propagate in varying directions.

Hyperbolic Tangent Function (Tanh)

The hyperbolic tangent function, a.k.a., the tanh function, is another type of activation functions. It is a smoother, zero-centered function having a range between -1 to 1. As a result, the output of the tanh function is represented by:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

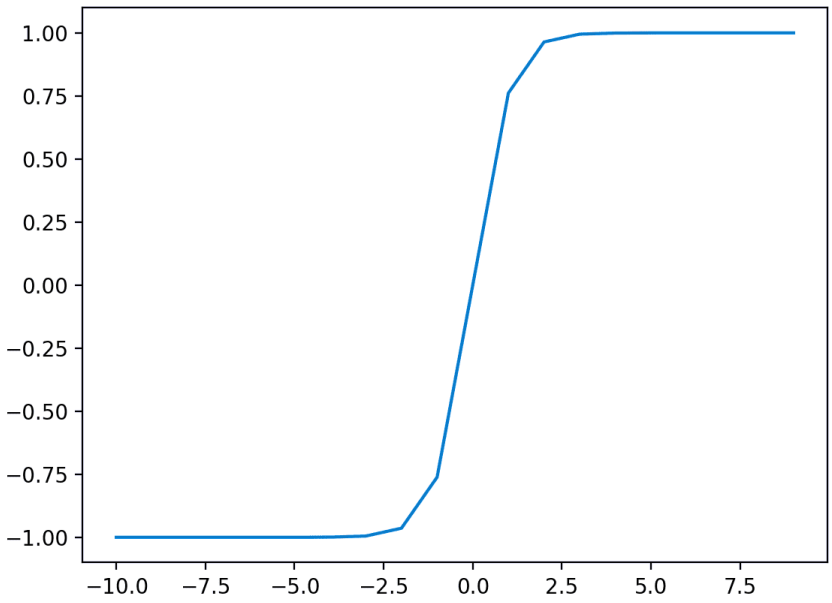


Figure 18: Tanh activation function.

The tanh function is much more extensively used than the sigmoid function since it delivers better training performance for multilayer neural networks. The biggest advantage of the tanh function is that it produces a zero-centered output, thereby supporting the backpropagation process. The tanh function has been mostly used in recurrent neural networks for natural language processing and speech recognition tasks.

However, the tanh function, too, has a limitation – just like the sigmoid function, it cannot solve the vanishing gradient problem. Also, the tanh function can only attain a gradient of 1 when the input value is 0 (x is zero). As a result, the function can produce some dead neurons during the computation process.

Softmax Function

The softmax function is another type of activation function used in neural networks to compute probability distribution from a vector of real numbers. This function generates an output that ranges between values 0 and 1 and with the sum of the probabilities being equal to 1. The softmax function is represented as follows:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

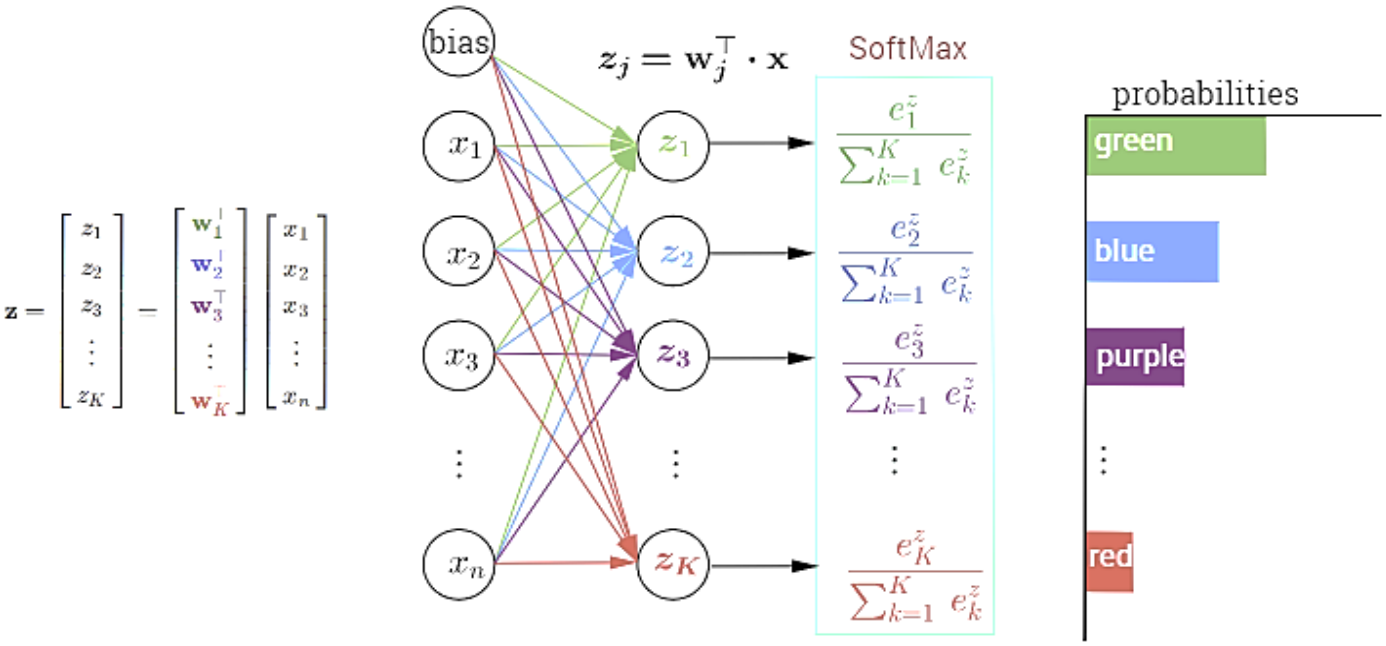


Figure 19: Multi-classes classification with NN and Softmax activation function.

This function is mainly used in multi-class models where it returns probabilities of each class, with the target class having the highest probability. It appears in almost all the output layers of the deep learning architecture where they are used. The primary difference between the sigmoid and softmax activation function is that while the former is used in binary classification, the latter is used for multivariate classification.

Advantages

- Softmax is optimal for maximum-likelihood estimation of the model parameters.
- The properties of softmax (all output values in the range (0, 1) and sum up to 1.0) make it suitable for a probabilistic interpretation that's very useful in machine learning.
- Softmax normalization is a way of reducing the influence of extreme values or outliers in the data without removing data points from the set.

How to Choose a Hidden Layer Activation Function

A neural network will almost always have the same activation function in all hidden layers. It is most unusual to vary the activation function through a network model. Traditionally, the sigmoid activation function was the default activation function in the 1990s. Perhaps through the mid to late 1990s to 2010s, the Tanh function was the default activation function for hidden layers.

The activation function used in hidden layers is typically chosen based on the type of neural network architecture. Modern neural network models with common architectures, such as MLP and CNN, will make use of the ReLU activation function, or extensions. If you're unsure which activation function to use for your network, try a few and compare the results.

The figure below summarizes how to choose an activation function for the hidden layers of your neural network model.

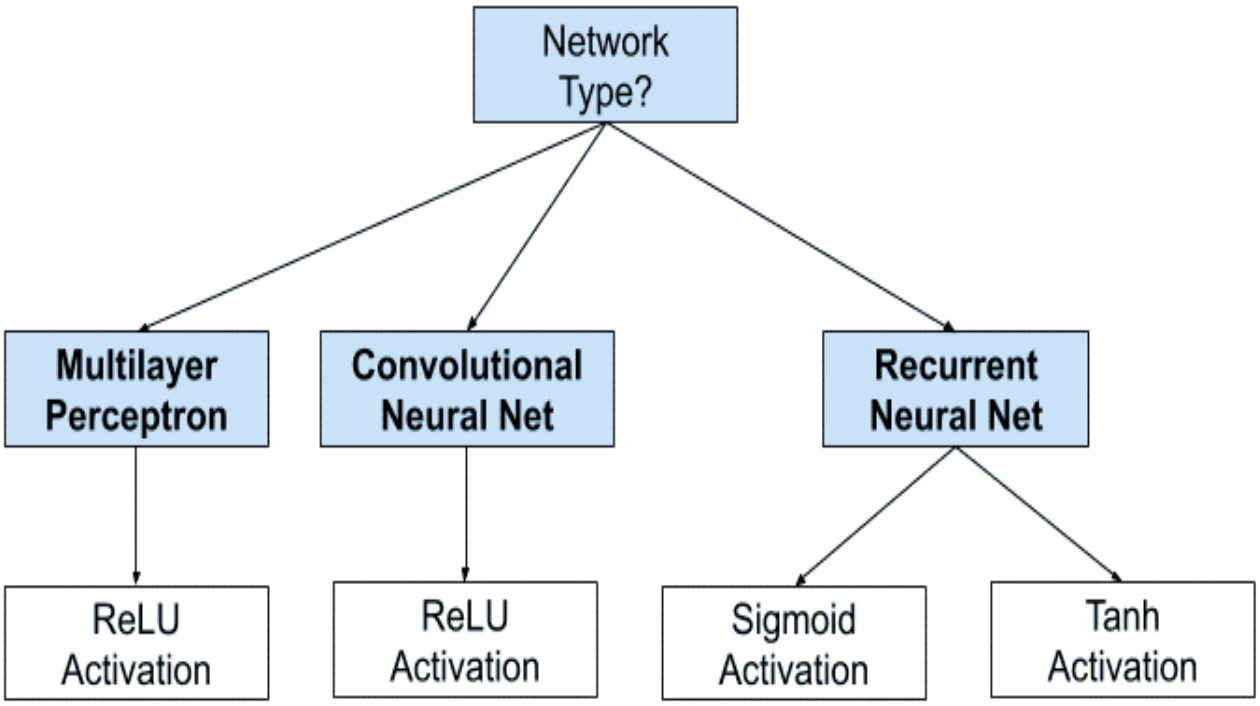


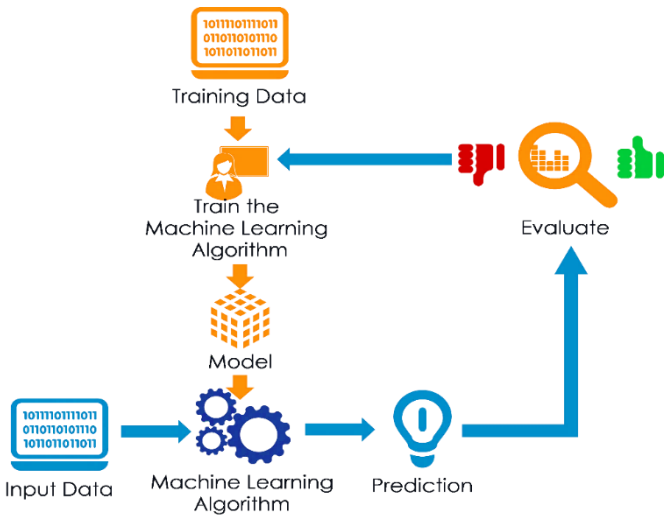
Figure 20: methods to Choose a Hidden Layer Activation Function

BUILDING THE DETECTOR MODEL

Given a dataset of 2D dashboard camera images, an algorithm needs to be developed to classify each driver's behavior and determine if they are driving attentively, wearing their seatbelt, or taking a selfie with their friends in the backseat etc...? This can then be used to automatically detect drivers engaging in distracted behaviors from dashboard cameras. Applying transfer learning to vision data using two different CNN architectures.

The following are the key contributions we make in this chapter:

- Using a publicly available distracted driver detection dataset to fine-tune the CNN model used to process vision data.
- Building a model to detect more distracted driving actions (ten) compared to most of the existing studies that aim to detect six or fewer actions.
- Analysis and evaluation the detection model results.



3.1 Dataset Description

The dataset consists of ten classes viz. safe driving, texting on mobile phones using right or left hand, talking on mobile phones using right or left hand, adjusting radio, eating, or drinking, hair and makeup, reaching behind and talking to passenger. Sample images of each class from the dataset are shown in figure 20. The data was collected from thirty-one participants from seven different countries using four different cars and incorporated several variations of the drivers and driving conditions. For example, drivers are exposed to different lighting conditions like sunlight and shadows. The dataset consists of 102,148 images divided into training set (22423) and test set (79725). We follow the same data distribution as in for true performance comparison.

Figure 21: An illustration of 10 categories of driving activities considered here. c0 - c9 represent the label of each image in the dataset.

3.2 Software Tools

3.2.1 Python 3.7.3

Adding Python to the implementation process has helped us to validate various ideas as it has many benefits as follows.

- Python's simple syntax allows developers to write codes that are reliable, concise, and readable.
- To help developers in the development process, several libraries and frameworks get involved.
- It basically means one can freely shift from one machine to another without making changes to the actual code (or with minimal changes). This functionality is allowed by python's framework.

3.2.2 Tensorflow 2.4.0 and Keras

TensorFlow is a powerful open-source software library developed by the Google Brain team for deep neural networks. the reasons of making TensorFlow special are:

- It works with all popular languages such as Python, C++, Java, R, and Go.
- Keras – a high-level neural network API that has been integrated with TensorFlow (in 2.0, Keras became the standard API for interacting with TensorFlow). This API specifies how software components should interact.
- TensorFlow allows model deployment and ease of use in production.
- Support for eager computation (see Chapter 2, TensorFlow 1.x and 2.x) has been introduced in TensorFlow 2.0, in addition to graph computation based on static graphs.

3.2.3 Sklearn

The sklearn.metrics module implements several loss, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values. Most implementations allow each sample to provide a weighted contribution to the overall score, through the *sample_weight* parameter.

In our project we use confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, and model_selection (shuffle the data and split them into the training set and the validation set).

3.2.4 NumPy and Pandas

NumPy provides the essential multi-dimensional array-oriented computing functionalities designed for high-level mathematical functions and scientific computation.

Like NumPy, Pandas is one of the most widely used python libraries in data science. It provides high-performance, easy to use structures and data analysis tools. Unlike NumPy library which provides objects for multi-dimensional arrays, Pandas provides in-memory 2d table object called Dataframe. It is like a spreadsheet with column names and row labels.

3.2.5 Seaborn and Matplotlib

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

3.2.6 Pickle

Pickle is used for serializing and de-serializing Python object structures, also called marshalling or flattening. It is useful for storing data.

3.2.7 OpenCV

It’s an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception.

3.3 Hardware Components

3.3.1 NVIDIA Geforce GTX1060 GPU, CUDA, CuDNN

The NVIDIA CUDA® Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks. cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers.

Deep learning researchers and framework developers worldwide rely on cuDNN for high-performance GPU acceleration. It allows them to focus on training neural networks and developing software applications rather than spending time on low-level GPU performance tuning. cuDNN accelerates widely used deep learning frameworks, including Caffe2, Chainer, Keras, MATLAB, MxNet, PaddlePaddle, PyTorch, and TensorFlow. For access to NVIDIA optimized deep learning framework containers that have cuDNN integrated into frameworks, visit NVIDIA GPU CLOUD to learn more and get started.

3.3.2 Raspberry Pi 4B (2GB)

The Pi 4 can be used to run basic ML and AI tasks with its built in camera input. This can be used for image recognition at the edge, for basic tasks such as seeing movement, recognizing objects and running basic inference tasks. Code can also be compiled faster owing to the faster CPU and RAM, allowing for better load and run times for algorithms. Machine learning tasks in frameworks such as Scikit-learn also perform twice as much better on the Pi 4 as it is a vastly more powerful PC than its previous iterations. Along with the dual HDMI output, the Pi can also be used as a coding station that can be picked up and used in any situation where a monitor is available. Owing to the power draw, a high-voltage power bank can be used for power at the edge, allowing for a truly portable experience.

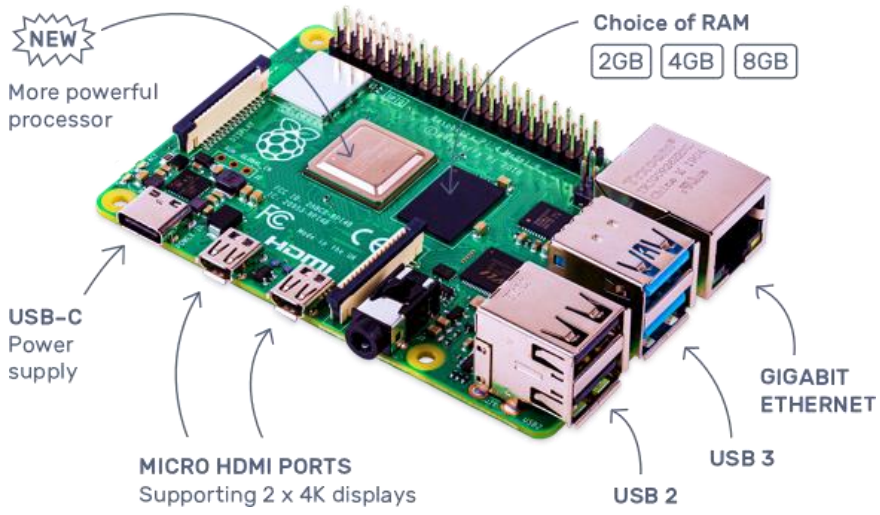


Figure 22: Raspberry pi 4 model B.

3.3.3 Raspberry Pi Camera Module 1.3 (5MP)

The Raspberry Pi Camera Board plugs directly into the CSI connector on the Raspberry Pi. It's able to deliver a crystal clear 5MP resolution image, or 1080p HD video recording at 30fps! Latest Version 1.3! Custom designed and manufactured by the Raspberry Pi Foundation in the UK, the Raspberry Pi Camera Board features a 5MP (2592 x 1944 pixels) Omni vision 5647 sensor in a fixed focus module. The module attaches to Raspberry Pi, by way of a 15 Pin Ribbon Cable, to the dedicated 15-pin MIPI Camera Serial Interface (CSI), which was designed especially for interfacing to cameras. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data to the BCM2835 processor. The board itself is tiny, at around 25mm x 20mm x 9mm, and weighs just over 3g, making it perfect for mobile or other applications where size and weight are important. The sensor itself has a native resolution of 5 megapixel, and has a fixed focus lens onboard. In terms of still images, the camera is capable of 2592 x 1944 pixel

static images, and also supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 video recording. The camera is supported in the latest version of Raspbian, the Raspberry Pi's preferred operating system.

3.3.4 C Charger (5v, 3Amp)

Power supply 5V 3A fast rapid charge AC adapter w/ 1.5m extra-long on off power switch micro-USB cable for Raspberry Pi 4 Model C.

3.3.5 The Raspberry Pi Camera Board Features

- Fully Compatible with Both the Model A and Model B Raspberry Pi.
- 5MP Omni Vision 5647 Camera Module.
- Still Picture Resolution: 2592 x 1944.
- Video: Supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 Recording.
- 15-pin MIPI Camera Serial Interface - Plugs Directly into the Raspberry Pi Board.
- Size: 20 x 25 x 9 mm and weight 3g.
- Fully compatible with many Raspberry Pi cases.

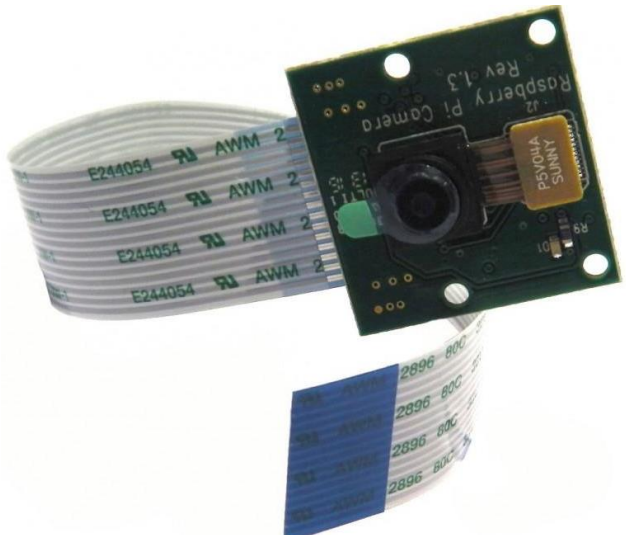


Figure 23: Raspberry Pi Camera board.

3.3.6 5” LCD Touch Screen (800x480)

Designed for Raspberry Pi, the 5-inch HDMI display screen monitor holds a resistive touch screen with a resolution of 800x480. It supports back-light control with a button on the backside to control. With the HDMI interface, it doesn’t occupy any I/O port of Raspberry Pi. This touch screen monitor supports touch screen control and 800 x480 resolution, providing a perfect visual experience for you.

Figure 24: 5” LCD Touch Screen (800x480).

3.3.7 Buzzer

Alert the driver when a distraction occurred.

3.4 Image Preprocessing

Each image needs to be preprocessed before going to the classifier, and the float chart of image preprocessing is shown in figure 21. Each image is first converted into a high-dimensional 640 × 480 × 3 matrix based on its RGB values on each pixel, then resized to a 224 x 224 x 1 matrix using CV2 in order to improve the computing efficiency of classifier, and finally flattened into a vector. For each flattened vector, a numerical label in the range of 0-9 is assigned based on the category it belongs.

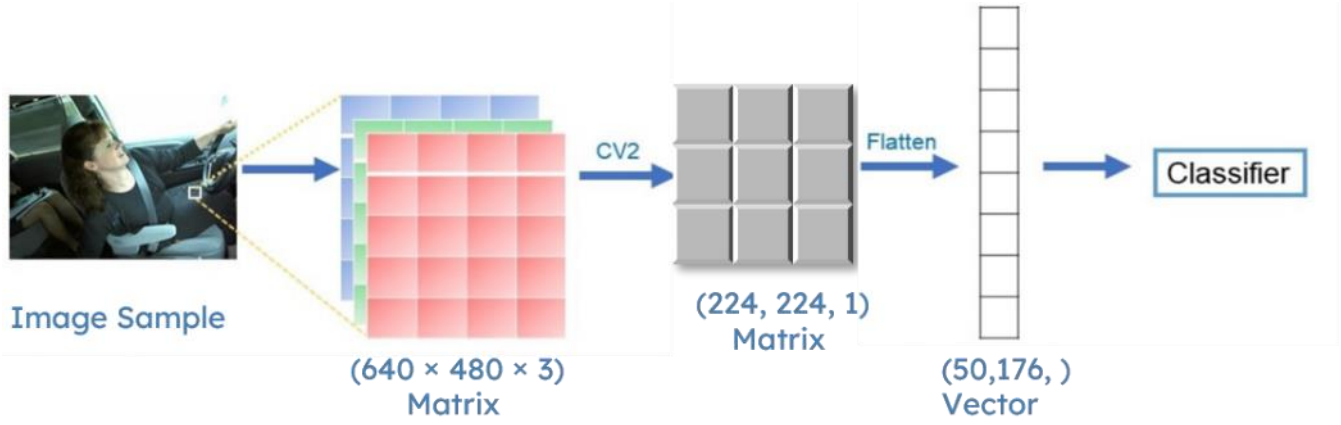


Figure 25: Flow chart image processing.

As for the whole training data, we shuffle the data and split them into the training set and the validation set by 80% / 20% and pile each training example along the column axis. Thus, the final training set matrix xtrain has the dimension (17939, 12288), the final training label vector yvalid has the dimension (17939,), the final validation set matrix xvalid has the dimension (4485, 12288), the final training label vector yvalid has the dimension (4485,).

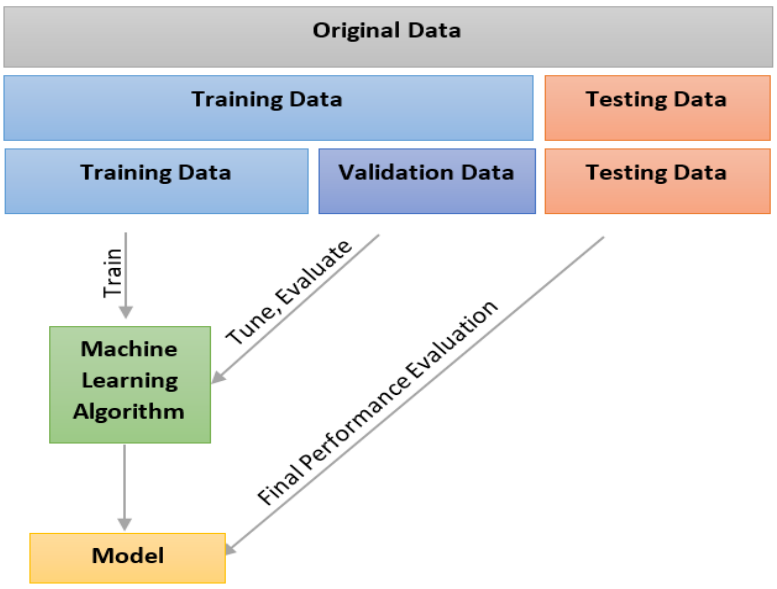


Figure 26: Splitting dataset.

3.5 Experiments

3.5.1 First Model - Using Transfer learning and fine tuning

Choose the suitable Model

Most of the time you won't want to train a whole convolutional network yourself. Modern ConvNets training on huge datasets like ImageNet take weeks on multiple GPUs. Instead, most people use a pretrained network either as a fixed feature extractor, or as an initial network to fine tune.

Using VGG16 trained on the ImageNet dataset as a feature extractor. Below is a diagram of the VGGNet architecture, with a series of convolutional and MaxPooling layers, then three fully connected layers at the end that classify the 1000 classes found in the ImageNet database. The idea here is that we keep all the convolutional layers but replace the final fully connected layer with our own classifier. This way we can use VGGNet as a fixed feature extractor for our images then easily train a simple classifier on top of that.

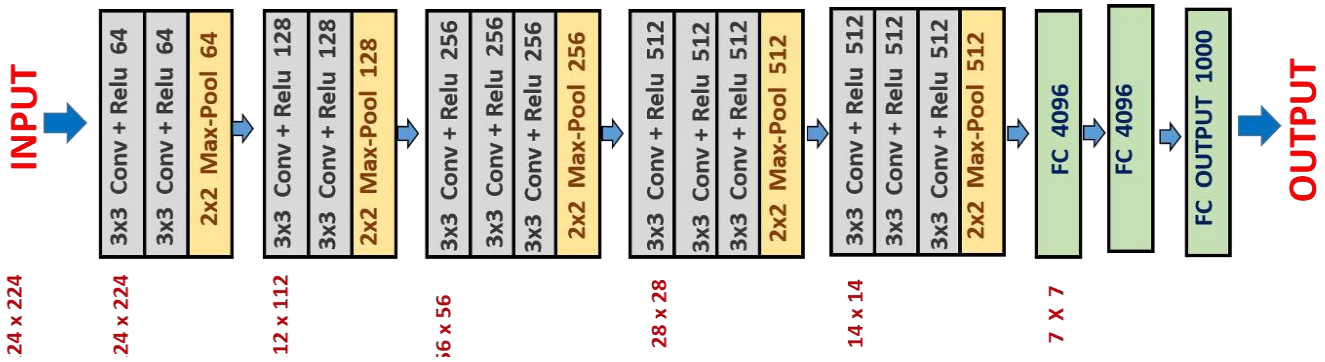


Figure 27: VGG16 Architecture.

Initialize Hyper Parameters

Table 2: Fine tuning model hyper-parameters.

Parameter	Image size	Color type	# Of Classes	# Of batches	# Of epochs	Learning rate	Beta
Value	224x224	RGB	10	50	30	0.01	0.9

3.5.2 Preprocessing And Load Data Images

Load training and testing datasets

Using OpenCV and tqdm Libraries we loaded our data to use it in training and testing.

Convert data to batches of tensors.

Generate batches of tensor image data with real-time data augmentation. So, the data will be looped over (in batches).

Preparing Image Augmentation and Normalization

Since our training image set had only ~22K images, we wanted to synthetically get more images from the training set to make sure the models don't overfit as the neural nets have millions of parameters. Image Augmentation is a technique that creates more images from the original by performing actions such as shifting width and/or height, rotation, and zoom. In our project, Image Augmentation had a few additional advantages. Sometimes, the difference between images from the two different classes can be very subtle. In such cases, getting multiple looks at the same image through different angles will help. If you look at the images below, we see that they are almost similar but in the first picture the class is 'Talking on the Phone — Right' and the second picture belongs to the 'Hair and Makeup' class.

Table 3: Fine tuning model data shapes.

Dataset	Training Images	# Of training Images	Validation Images	# Of validation Images
Shape	(19060, 224, 224, 3)	196060	(3364, 224, 224, 3)	3364

Define the Model

To maximize the value from transfer learning, we added a few extra layers to help the model adapt to our use case. we added an AvaregePooling, a Dense layer with 1024 units, and a Dense SoftMax layer of 10 units. We used ReLU activation for the dense layer of 1024 units so that I stop forwarding negative values through the network. I use a 10 units dense layer in the end with SoftMax activation as I have 10 classes to predict from in the end which distraction the driver made. The SoftMax layer will output the value between (0,1) based on the confidence of the model that which class the images belongs to. After the creation of SoftMax layer the model is finally prepared. Now I need to compile the model.

we should train only the extra layers added to the pre-existing architecture. Naturally, we started by using the ImageNet weights and trained only the new layers since the number of parameters to train would be lesser and the model would train faster.

So the model architecture is created as shown:

```
Model 1 network...
Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.
1/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 5s 0us/step
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, None, None, 3)     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, None, None, 64)    1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, None, None, 64)    36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, None, None, 64)    0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, None, None, 128)   73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, None, None, 128)   147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, None, None, 128)   0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, None, None, 256)   295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, None, None, 256)   590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, None, None, 256)   590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, None, None, 256)   0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, None, None, 512)   1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, None, None, 512)   0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, None, None, 512)   0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              525312
_________________________________________________________________
dense_2 (Dense)              (None, 10)                10250
=================================================================
Total params: 15,250,250
Trainable params: 535,562
Non-trainable params: 14,714,688
_________________________________________________________________
```

Compiling Model

Here I will be using RMSprop optimizer to reach to the global minima while training out model. If I am stuck in local minima while training, then the RMSprop optimizer will help us to get out of local minima and reach global minima. We will also specify the learning rate of the optimizer, using the default learning rate. If our training is bouncing a lot on epochs, then we need to decrease the learning rate so that we can reach global minima.

After the creation of the model will import ModelCheckpoint and EarlyStopping method from Keras. I will create an object of both and pass that as callback functions to fit_generator.

Re-train the Model

After the creation of the model will import ModelCheckpoint and EarlyStopping method from Keras. I will create an object of both and pass that as callback functions to fit_generator.

ModelCheckpoint helps us to save the model by monitoring a specific parameter of the model. In this case I am monitoring validation accuracy by passing validation accuracy to ModelCheckpoint. The model will only be saved to disk if the validation accuracy of the model in current epoch is greater than what it was in the last epoch.

EarlyStopping helps us to stop the training of the model early if there is no increase in the parameter which I have set to monitor in EarlyStopping. In this case I am monitoring validation accuracy by passing validation accuracy to EarlyStopping. I have here set patience to 6 which means that the model will stop to train if it doesn't see any rise in validation accuracy in 6 epochs.

I am using model.fit_generator as I am using ImageDataGenerator to pass data to the model. I will pass train and test data to fit_generator. In the fitting process steps_per_epoch will set the batch size to pass training data to the model and validation_steps will do the same for test data. You can tweak it based on your system specifications.

After the training process we get a validation loss as shown.

```
Epoch 00005: val_loss did not improve from 1.74111
Epoch 6/30
358/358 [==============================] - 313s 874ms/step - loss: 1.4257 - acc: 0.5058
- val_loss: 1.2569 - val_acc: 0.5462

Epoch 00006: val_loss did not improve from 1.74111
Epoch 00006: early stopping
```

Results

Once we have trained the model, we visualized training/validation accuracy and loss using Matplotlib library. As you may have noticed I am passing the output of mode.fit_generator to hist variable. All the training/validation accuracy and loss are stored in hist and I will visualize it from there. The pre-trained ImageNet model is used for weight initialization and concept of transfer learning is applied. Weights of all the layers of network are updated the dataset. After rigorous experimentation, all the hyperparameters are finetuned.

We saw that the accuracy on validation set plateaued at **55%**. Hence, we decided to make another experiment with another CNN architecture. The other reason that forced us to change the method is that the accuracy in the validation set that is greater than it in the training set.

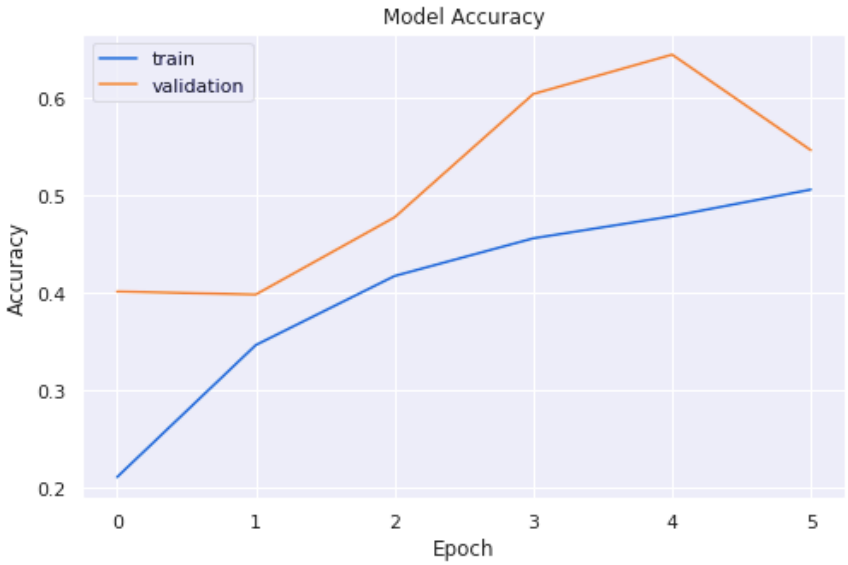


Figure 28: Fine tuning model accuracy.

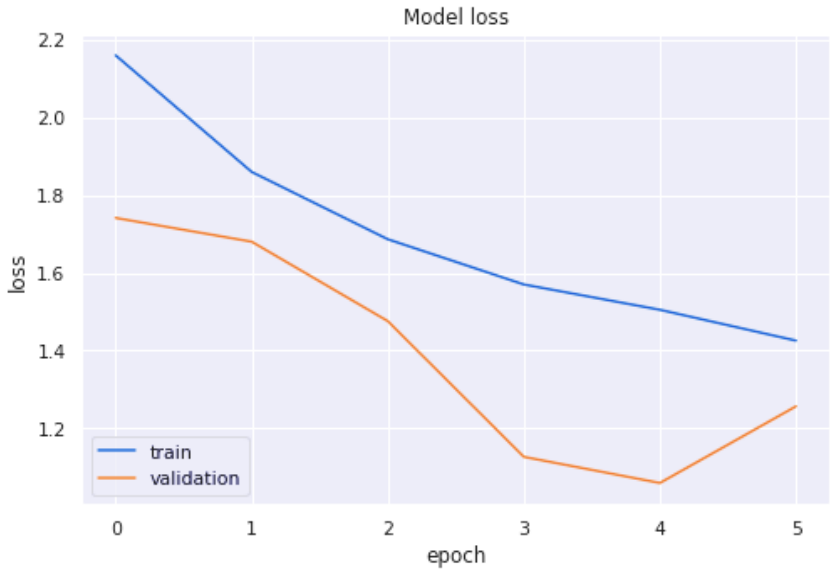


Figure 29: Fine tuning model losses

3.5.3 Second Model - Build CNN Architectures From Scratch

Define The Model

First, we make a sheet containing the dimensions, activation shapes and sizes for the architecture just as shown below.

Layers	AF	Filters	Stride	Filter size	Padding	Output shape	Parameters
Input image	-	-	-	-	-	(#, 224, 224, 1)	50,176
Conv2D_1	ReLU	64	1	(2, 2)	same	(#, 224, 224, 64)	320
MaxPooling_1	-	-	2	(2, 2)	-	(#, 112, 112, 64)	0
Conv2D_2	ReLU	128	1	(2, 2)	same	(#, 112, 112, 128)	32,896
MaxPooling_2	-	-	2	(2, 2)	-	(#, 56, 56, 128)	0
Conv2D_3	ReLU	256	1	(2, 2)	same	(#, 56, 56, 256)	131,328
MaxPooling_3	-	-	2	(2, 2)	-	(#, 28, 28, 256)	0
Conv2D_4	ReLU	512	1	(2, 2)	same	(#, 28, 28, 512)	524,800
MaxPooling_4	-	-	2	(2, 2)	-	(#, 14, 14, 512)	0
Dropout	-	-	-	-	-	(#, 14, 14, 512)	0
Flatten	-	-	-	-	-	(#, 100352)	0
Dense	ReLU	-	-	-	-	(#, 500)	50,176,500
Dropout	-	-	-	-	-	(#, 500)	0
Dense	Softmax	-	-	-	-	(#, 100)	5010
Total params: 50,870,854 **Trainable params:** 50,870,854 **Non-trainable params:** 0							

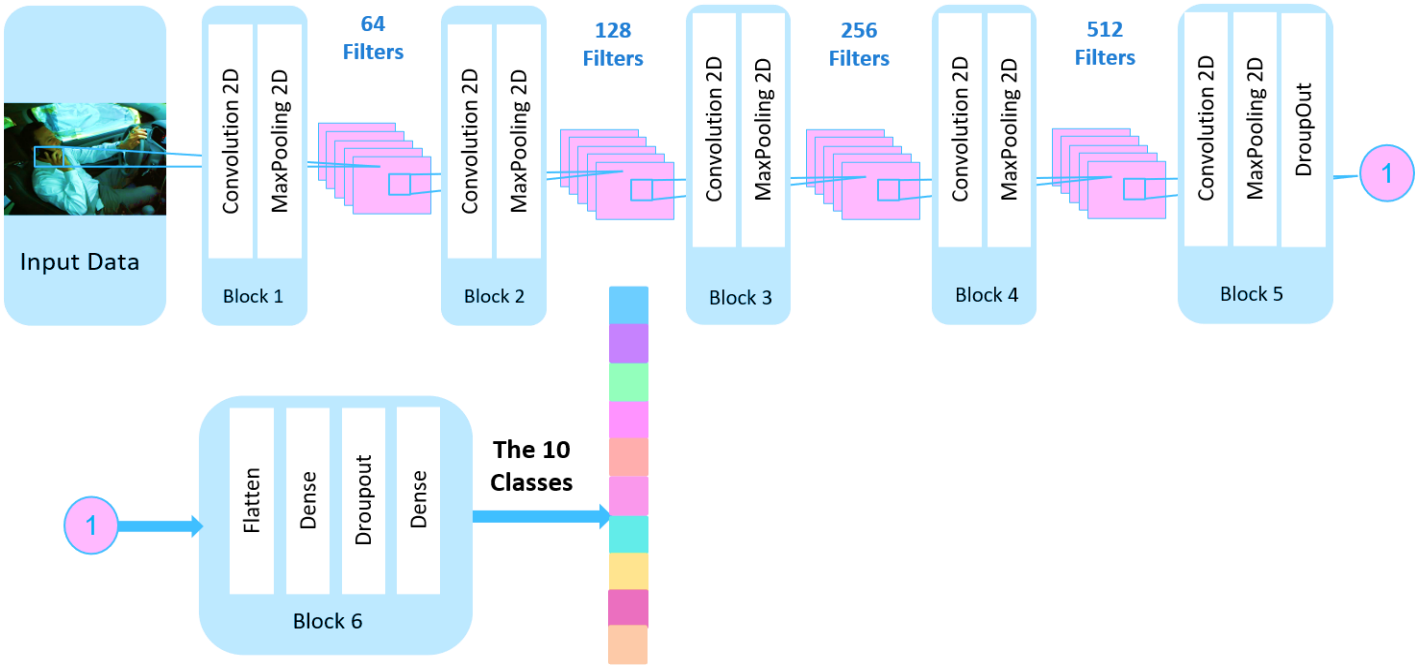


Figure 30: Our CNN architecture model.

Convolution Layers

the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image of a particular size (N×N) and a filter of a particular size (m×m). By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (m×m). The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image of a size (N−m+1)/s × (N−m+1)/s, as ‘s’ is the stride.

MaxPooling Layers

A Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations. In Max Pooling, the largest element is taken from feature map.

A Fully Connected layer (Flatten Layer)

It consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture. In this, the input image from the previous layers is flattened and fed to the FC layer.

The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place.

Dense Layer

The dense layer is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer. The dense layer is found to be the most used layer in the models. The output generated by the dense layer is an 'm' dimensional vector. Thus, dense layer is basically used for changing the dimensions of the vector. Dense layers also apply operations like rotation, scaling, translation on the vector.

Dropout Layer

Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on a new data. To overcome this problem, a dropout layer is utilized wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network.

Initialize The Hyper-Parameters and Preparing Data

Table 4: CNN model hyperparameters.

Parameter	Image size	Color type	# Of Classes	# Of batches	# Of epochs	Learning rate	Beta
Value	224x224	Grayscale	10	40	25	0.01	0.9

All the hyper-parameters are finetuned.

Table 5:CNN Model data shape.

Dataset	Training Images	# Of training Images	Validation Images	# Of validation Images
Shape	(17939, 224, 224, 1)	17939	(4485, 224, 224, 1)	4485

Compiling the Model

Choose the cost function.

We used Categorical Cross-Entropy loss, also called SoftMax Loss. It is a SoftMax activation plus a Cross-Entropy loss. If we use this loss, we will train a CNN to output a probability over the more than class. It is

used for multi-class classification. Discarding the elements of the summation which are zero due to target labels, we can write:

$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

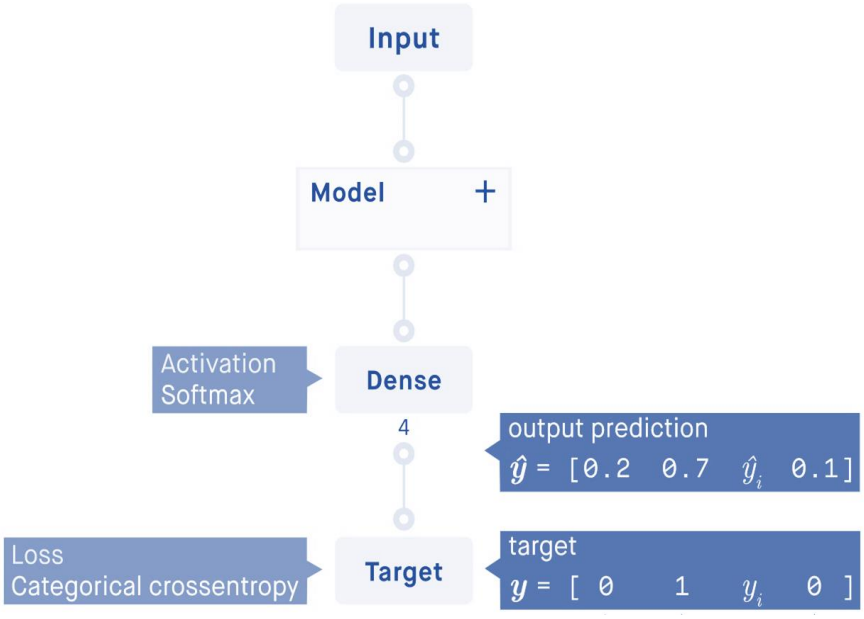


Figure 31: The Categorical Cross-Entropy cost function.

Choose The Optimizer

As optimizers minimize an objective function parameterized by a model's parameters by updating the parameters in the opposite direction of the gradient of the objective function to the parameters.

We used RMSprop optimizer as it similar to the gradient descent algorithm, but it has momentum. The RMSprop optimizer restricts the oscillations in the vertical direction. Therefore, we can increase our learning rate and our algorithm could take larger steps in the horizontal direction converging faster. The difference between RMSprop and gradient descent is on how the gradients are calculated. The following equations show how the gradients are calculated for the RMSprop and gradient descent with momentum. The value of momentum is denoted by beta and is usually set to 0.9.Sometimes the value of *vdw* could be really close to 0. Then, the value of our weights could blow up. To prevent the gradients from blowing up, we include a parameter epsilon in the denominator which is set to a small value.

$$v_{dw} = \beta \cdot v_{dw} + (1-\beta) \cdot dw^2$$

$$v_{db} = \beta \cdot v_{dw} + (1-\beta) \cdot db^2$$

$$W = W - \alpha \cdot \frac{dw}{\sqrt{v_{dw}} + \epsilon}$$

$$b = b - \alpha \cdot \frac{db}{\sqrt{v_{db}} + \epsilon}$$

$$v_{dw} = \beta \cdot v_{dw} + (1-\beta) \cdot dw$$

$$v_{db} = \beta \cdot v_{dw} + (1-\beta) \cdot db$$

$$W = W - \alpha \cdot v_{dw}$$

$$b = b - \alpha \cdot v_{db}$$

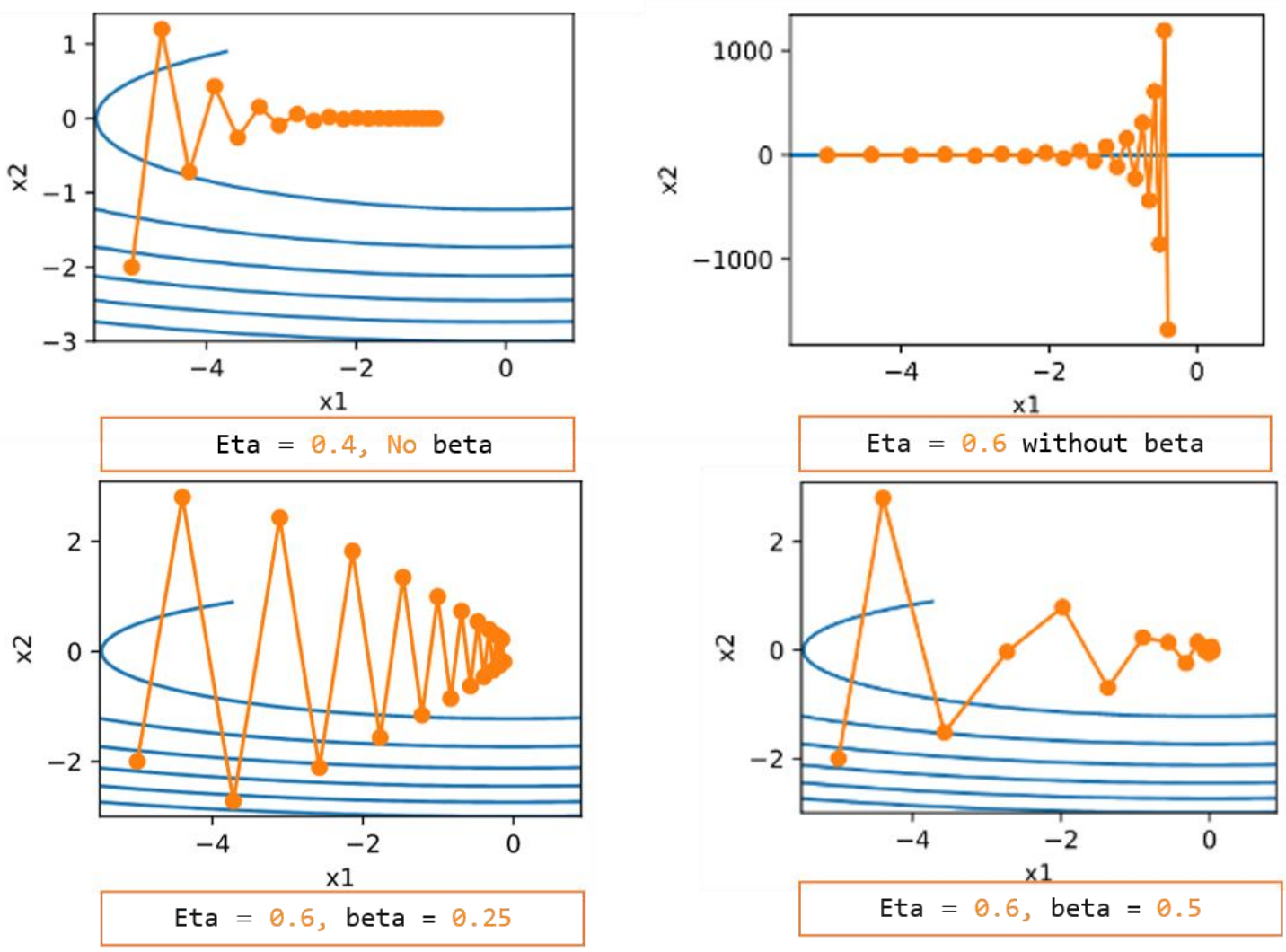


Figure 32: Changing the training time over the change of Eta and Beta.

Create callback.

ModelCheckpoint callback is used in conjunction with training to save a model or weights (in a checkpoint file) at some interval, so the model or weights can be loaded later to continue the training from the state saved.

Learnings/Training

In CNN, basic features such as lines, edges, and colours are learnt, and these features are merged to detect higher-order features. CNN architecture consists of many layers that are designed for individual tasks, but we can summarize them as feature learning and classification stages. In addition to the general structure of CNN, batch normalization and dropout techniques used for regularization in CNN architectures should be mentioned. We use both techniques in our work. The concept of normalization is commonly used in the machine learning area and aims to speed up the training process and eliminate excessive fluctuations by

scaling input values. Batch normalization is performed similarly. However, since the input data is processed as batches, normalization is also carried out for each batch. Especially in CNN applications, batch normalization is used not only in the input layer but also in the hidden layers. This prevents the weights and outputs from reaching extremely high or extremely low values throughout the entire network. The key goal of the dropout technique is to prevent over-fitting by ignoring some randomly chosen neurons in particular epochs during training. Thus, dropout forces the network to learn more robust features and helps to improve the generalization error. After 25 epochs the validation accuracy is 99.509%

Results And Model Evaluation

Accuracy score.

In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in dataset.

$$Accuracy = \frac{true\ positives + true\ negatives}{total\ eamples}$$

Precision score.

The precision is the ability of the classifier not to label as positive a sample that is negative. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative. The best value is 1 and the worst value is 0.

$$Precision = \frac{true\ positives}{true\ negatives + false\ positive}$$

Recall score.

The recall recall is the ability of the classifier to find all the positive samples. The recall is intuitively the ability of the classifier to find all the positive samples. The best value is 1 and the worst value is 0.

$$Recall = \frac{true\ positives}{true\ positive + false\ negatives}$$

F1 score

Also known as balanced F-score or F-measure. It can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:

$$F1 = 2\ \frac{precision\ x\ recall}{(precision + recall)}$$

In the multi-class and multi-label case, this is the average of the F1 score of each class with weighting depending on the average parameter.

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Figure 33: Evaluation metrics.

Table 6: The Model Results

Metric	Accuracy	Precision	Recall	F1 score
Result	0.995095	0.995112	0.995095	0.995091

The Model History Graphs.

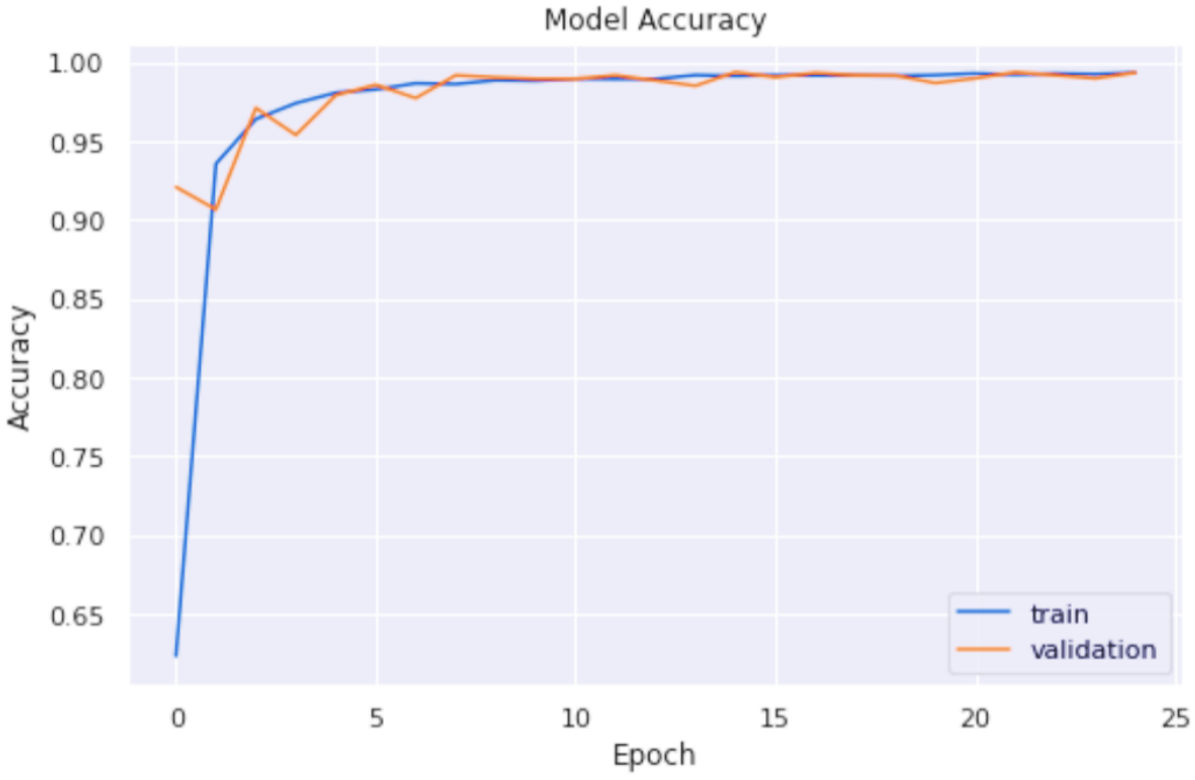


Figure 34: Train and validation accuracy.

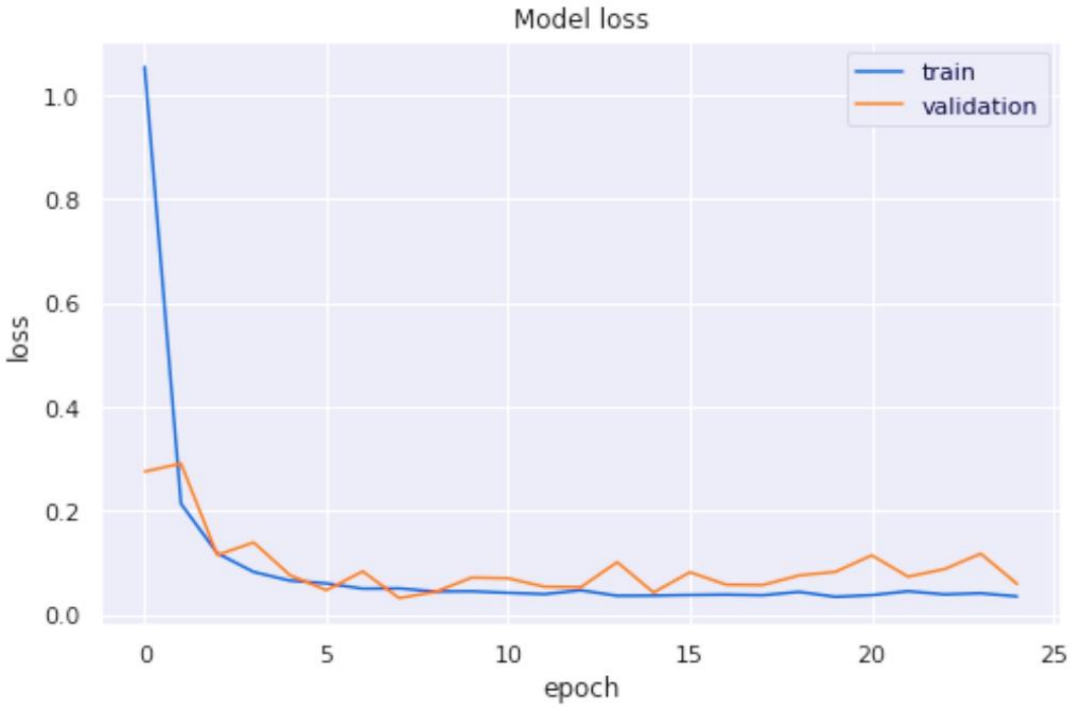


Figure 35: Train and validation losses.

Confusion matrix

The confusion matrix function evaluates classification accuracy by computing the confusion matrix with each row corresponding to the true class. The entry in a confusion matrix is the number of observations actually.

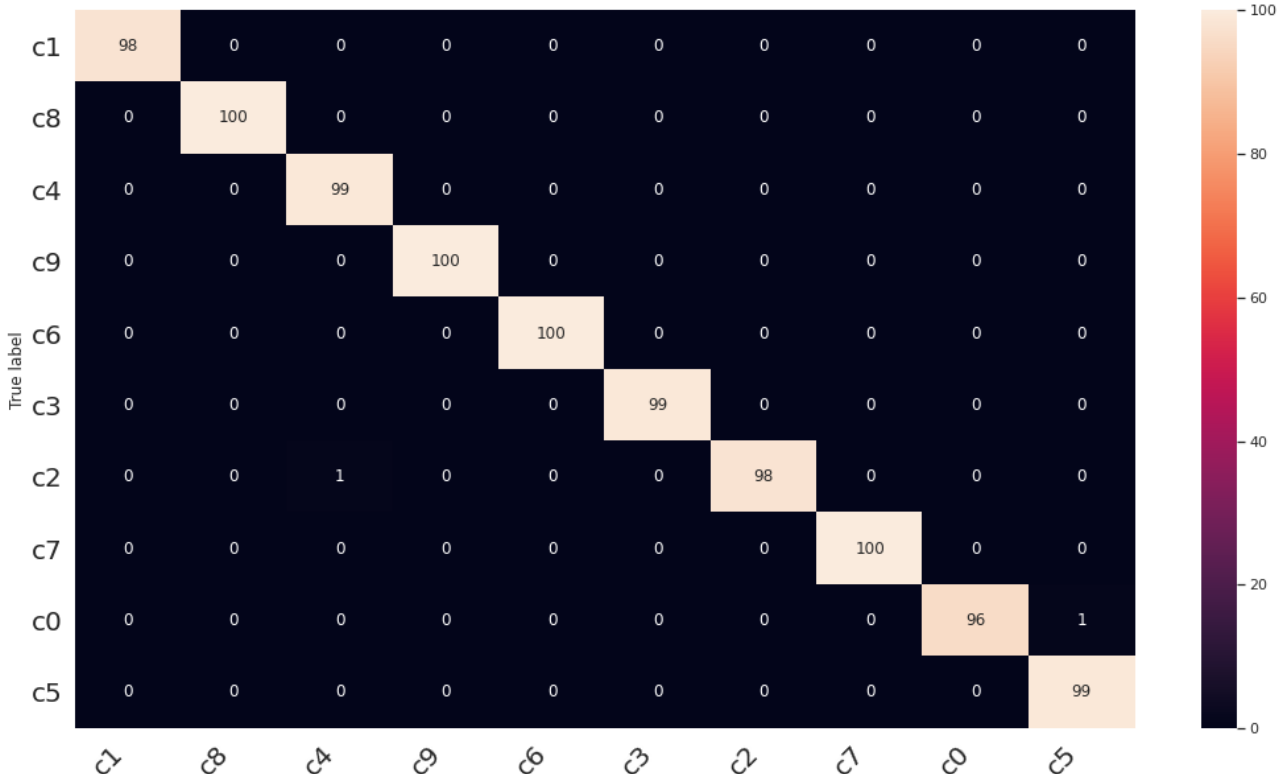


Figure 36: confusion matrix.

EXPERIMENTAL WORK

The connectivity between hardware components and collecting frames is an important point in our project, so we will discuss the following contributions in this chapter:

- connecting hardware components and setting up a suitable Python 3 environment on the Raspberry Pi by installing the required additional libraries for importing.
- capturing frames of the driver behavior using Pi camera.

4.1 Hardware Connection

We created a suitable python3 environment in Raspberry pi by installing the needed additional libraries for importing as the main python3 standard libraries are not enough for our code to work, so we need to find compatible libraries for our system and install them properly.

Some libraries might need additional libraries installed so they can function properly, the libraries that need to be installed for other libraries to function are called “dependencies”. Some libraries can be installed directly on terminal by using official site codes, others might need to be installed manually on virtual isolated environment due to reasons of incompatibility with python version or with the hardware.

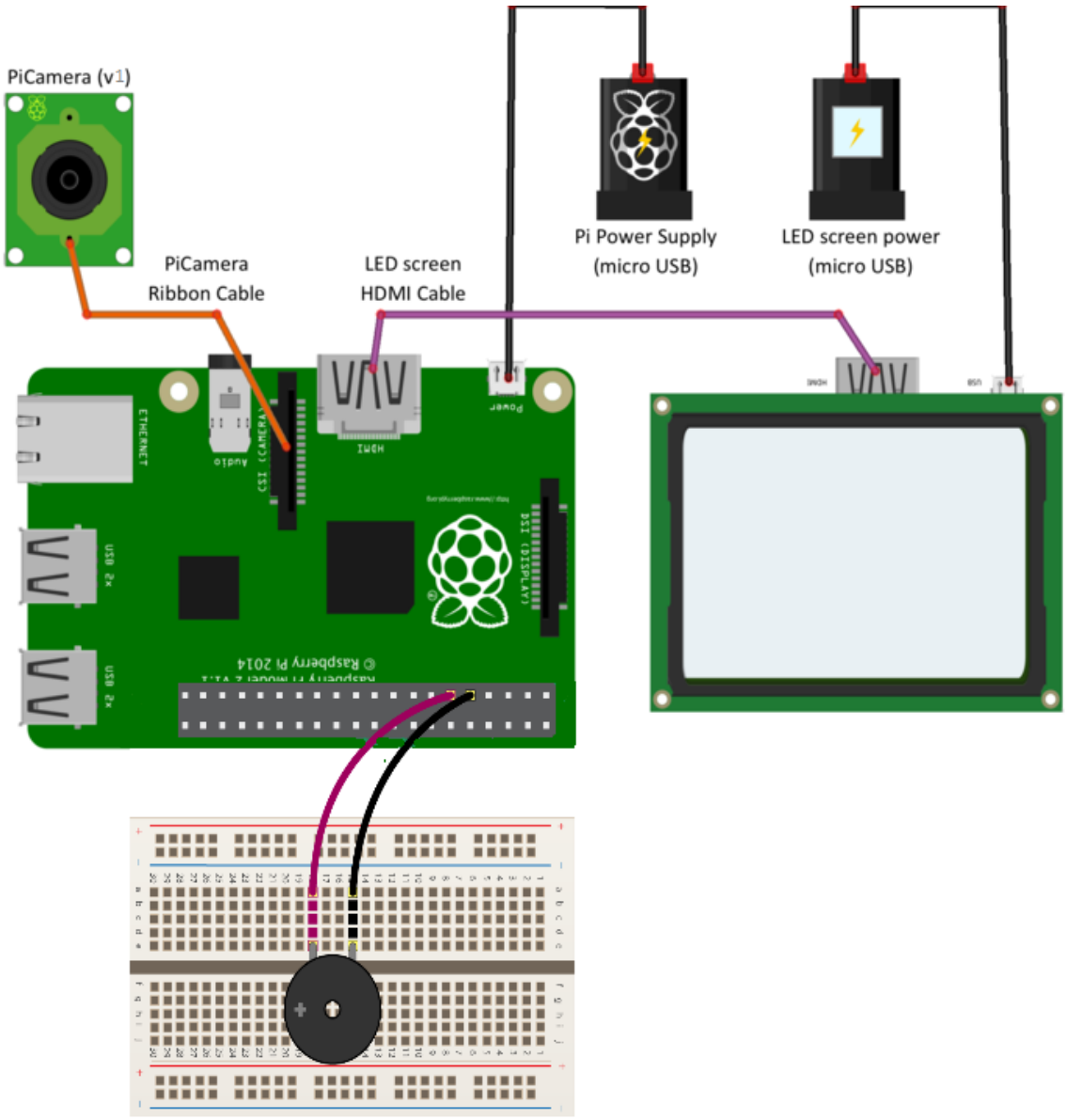


Figure 37: Hardware connections.

4.1.1 Embedded System

Since pi is commonly used in image processing application, it has compatible components ports within it and GPIO pins to connect to external components which we only needed to connect buzzer.

- Pi Camera is connected to camera port next to Ethernet port.
- Pi LCD is connected via microHDMI cable to HDMI port.
- Buzzer is powered by and connected via GPIO pins using test board.
- Pi is powered up by Type-c charger.
- Pi LCD is powered up by microUSB cable.

4.2 Raspbian

Raspbian is the official operating system that provides user interface for the raspberry pi which is a version of Linux Debian buster.

4.2.1 Why Raspbian and not RISC OS PI

Raspbian provides more tools for programming and is more compatible with the raspberry pi because it contains the official suite and has pip and python3 installed and allowed for regular updates which suites our project more.

While RISC is compatible with arm architecture hardware like raspberry pi and takes lighter space of storage it's more suitable for the use of running single application and doesn't provide a better environment for programming.

4.2.2 installation

1- Raspbian OS .iso is downloaded from official raspberry pi site.
2- The iso is burnt over microSD card to create a bootable device on PC.
3- MicroSD is inserted in its port and is booted and raspberry pi automatically installs the .iso file.

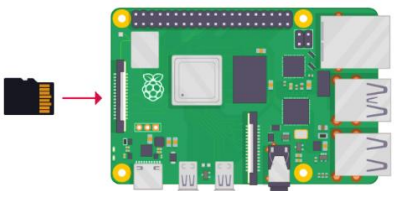


Figure 38: Raspbian set

4- Now the 32GB microSD acts as both storage for operating system and also storage for the to be installed libraries and it should be enough.
5- After booting Raspbian is installed and asks for user information like password and update.

4.2.3 Access Methods

It can be accessed directly via hdmi monitor device and it can be accessed remotely by many ways like via web browser over internet connection by accessing its local ip address or using a VNC viewer which also establish a connection between pi and PC over a network.

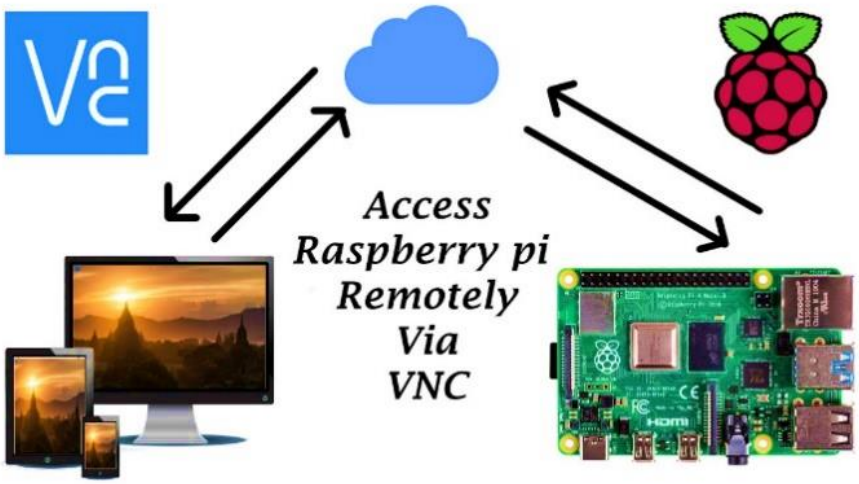


Figure 39: Access Methods

4.3 Libraries setup

4.3.1 Pip

It is the standard package manager for Python allows for installation and management of additional packages that are not part of the Python standard library via raspberry pi terminal.

all libraries are installed on main raspberry pi environment and TensorFlow is installed on a standalone virtual environment all through pip.

4.3.2 TensorFlow

We installed TensorFlow library on a virtual environment because, it gives us some more isolated space to experiment with, without 'damaging' the rest of our system, if we experiment a lot there is chance some dependencies could go in conflict, and it also allows us to switch between different versions

4.3.3 Virtualenv

This module provides support for creating lightweight “virtual environments” with their own site directories, optionally isolated from system site directories.

Each virtual environment can have its own independent set of installed python packages in its site directories and can be of different python versions.

We created a virtual environment on desktop and allowed it to use installed global libraries within raspberry pi main environment, and the main code is run from within the virtual environment.

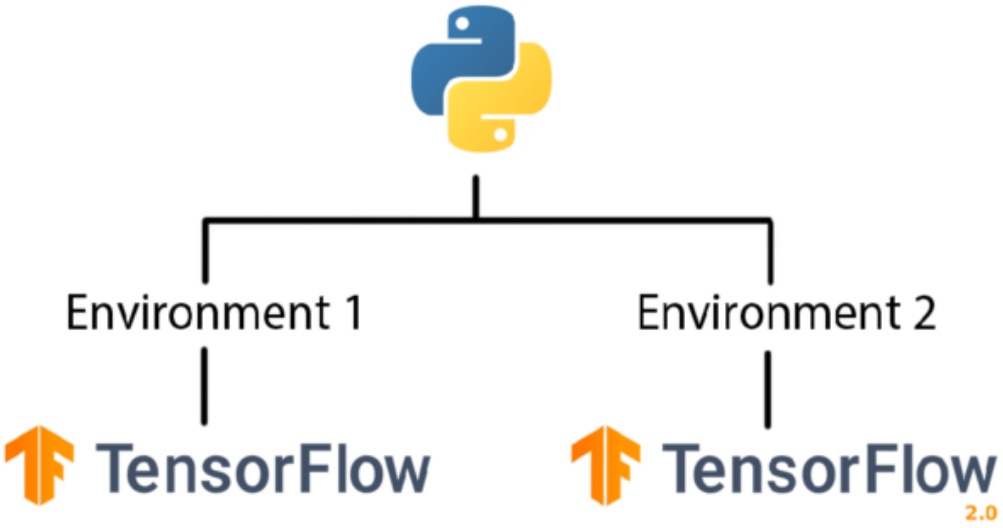


Figure 40: virtual environments

4.4 Capturing Frames

The goal is to keep capturing frames of the driver actions and store them in specific directory for passing them to the detector model then delete all these frames after the trip is being ended.

In order to achieve this goal two python libraries are needed:

- OpenCV
- OS.

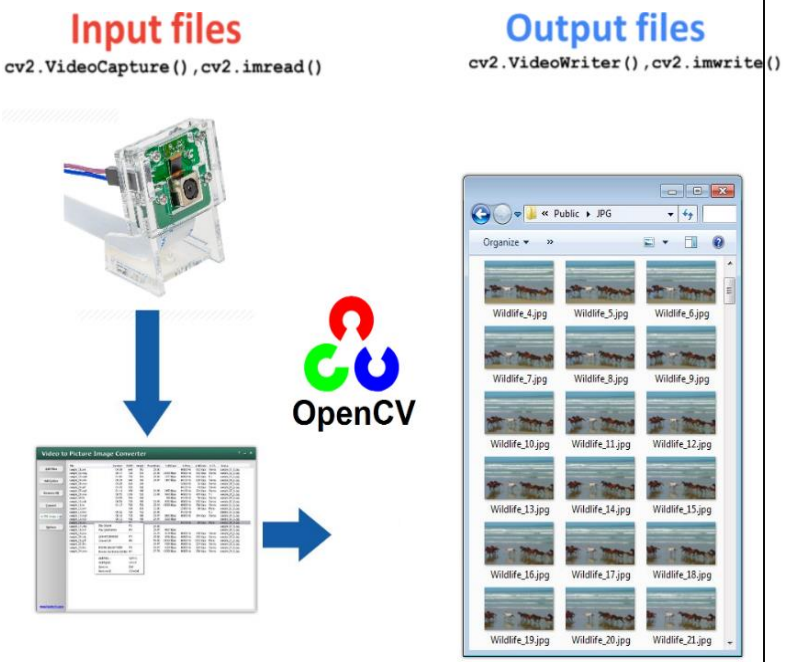


Figure 41: storing captured frames.

4.4.1 Capturing frame procedure

1. Create object `cam=cv2.VideoCapture(0)` to use first camera to capture video.
2. Checking whether a frame is captured or not, and also the show the captured frame using `cam.read()` function.
3. Record the number of captured frames in a variable, then reset the variable after the frames being deleted.
4. Create an infinite loop to keep the camera on until the stopped button is pressed by the driver.
5. We check if the button pressed or not every 200 milliseconds using `cv2.waitKey(200)` function.
6. Here we have two possible scenarios.
   a. Stopped button pressed – the program removes all the captured frames using `os.remove()` function.
   b. Stopped button not pressed – write a captured frame in directory numbered by the counter variable every 200ms using `cv2.imwrite()`.

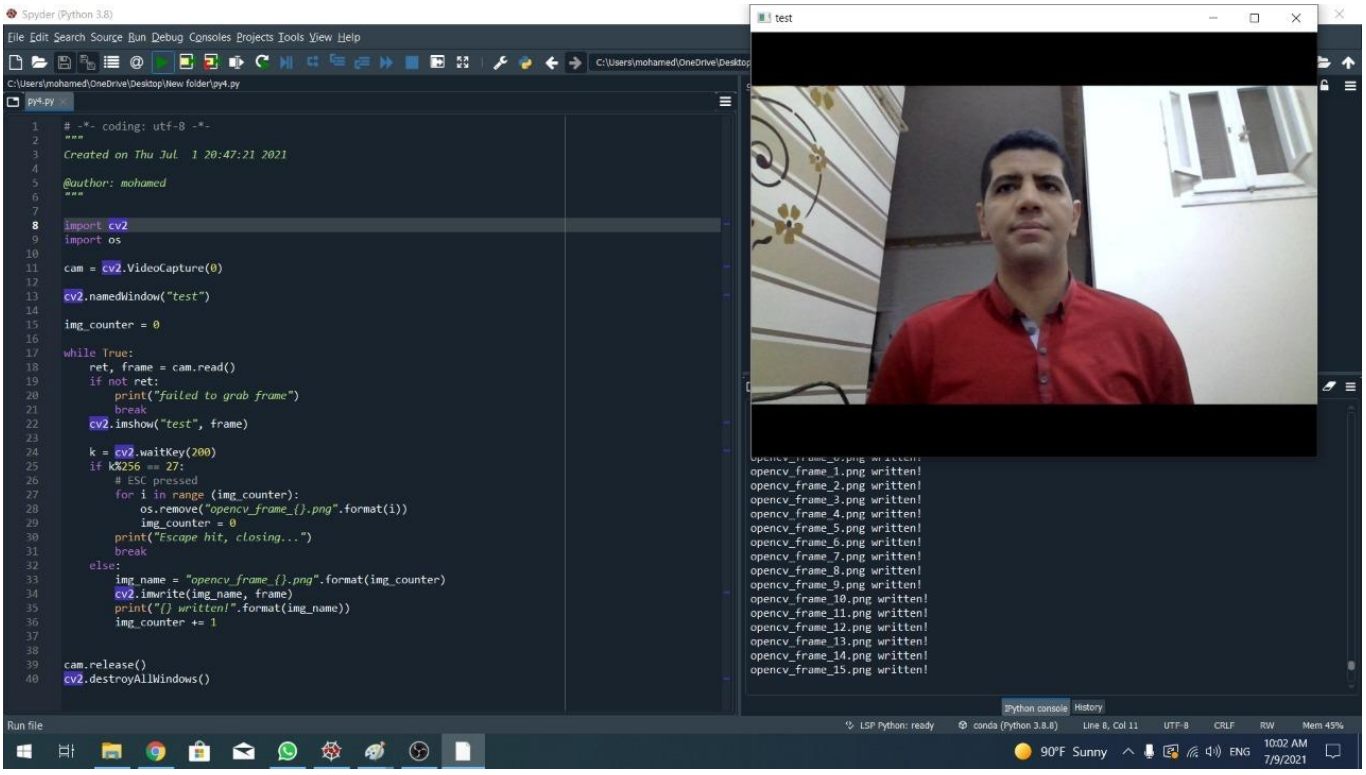


Figure 42: picamera while capturing frames.

4.4.2 Capturing frame Flow Chart

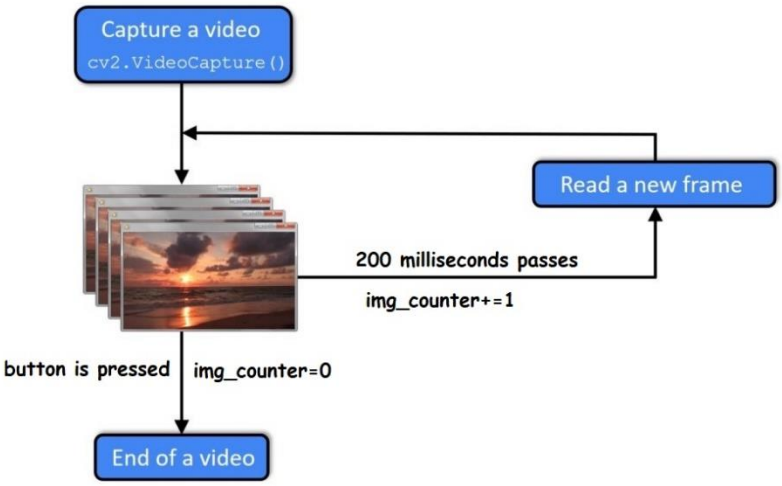


Figure 43: opencv path.

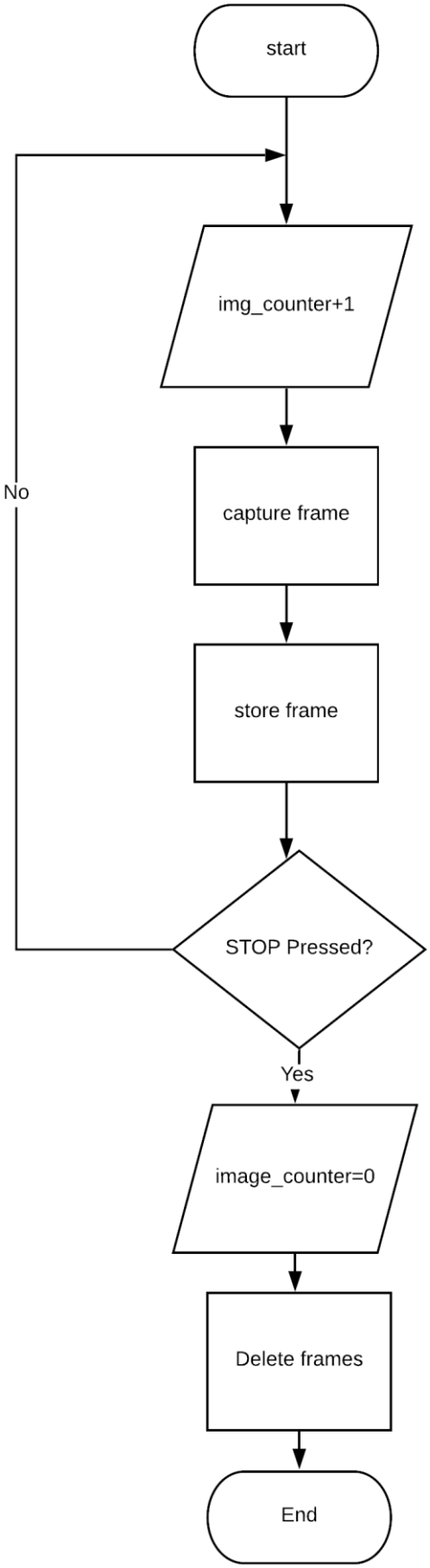


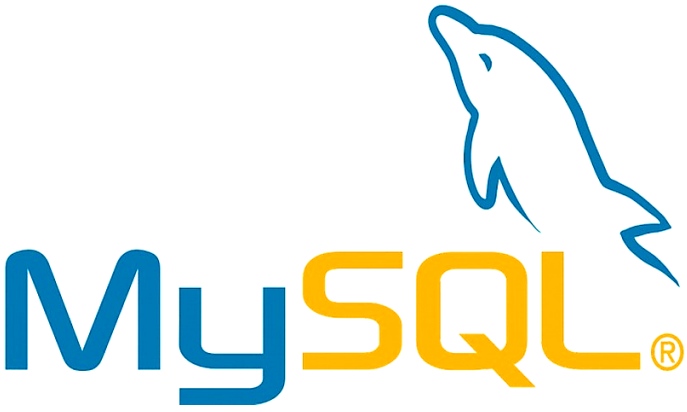
Figure 44: Pi camera driver flow chart

DATABASE AND GUI

Designing the visual composition and temporal behavior of a GUI is an important part of software application programming in the area of human–computer interaction. Its goal is to enhance the efficiency and ease of use for the underlying logical design of a stored program, a design discipline named usability. Methods of user-centered design are used to ensure that the visual language introduced in the design is well-tailored to the tasks.

The following are the key contributions we make in this chapter:

- Our GUI used to presents information to the user in the form of visual widgets that could be customized easily without the need for command codes.
- The python library Tkinter and its interface components (such as buttons, icons and menus).
- How we build a link between the database server and the python code, as well as how we capture distracted driving behaviors in a database.



5.1 Databases

5.1.1 Early History of Databases

Before databases existed, everything had to be recorded on paper. We had lists, journals, ledgers and endless archives containing hundreds of thousands or even millions of records contained in filing cabinets. When it was necessary to access one of these records, finding and physically obtaining the record was a slow and laborious task.

The database was created to try and solve these limitations of traditional paper-based information storage. In databases, the files are called records and the individual data elements in a record (for example, name, phone number, date of birth) are called fields.

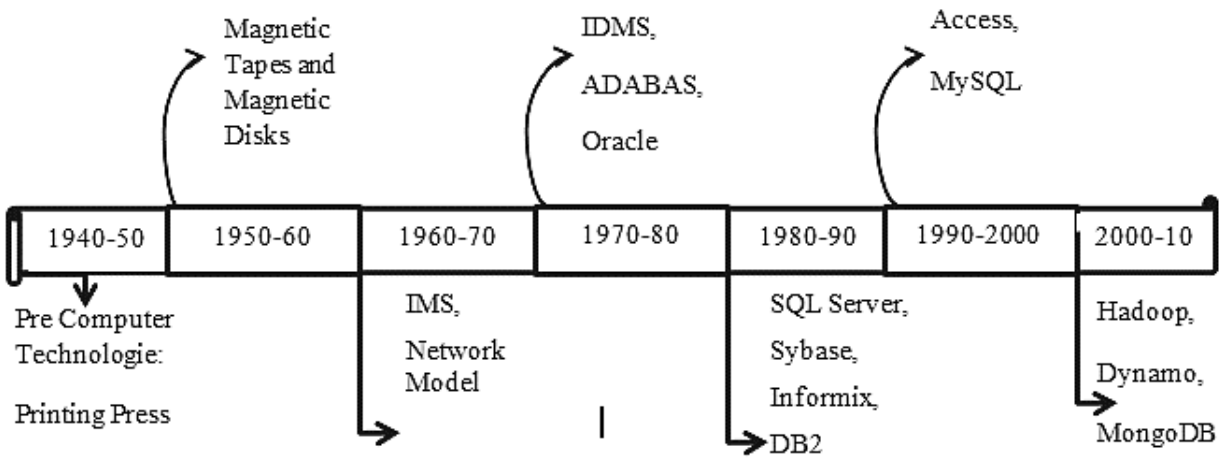


Figure 45: database timeline

5.1.2 Database Definition

A database is an organized collection of data, generally stored and accessed electronically from a computer system. It supports the storage and manipulation of data. In other words, databases are used by an organization as a method of storing, managing and retrieving information.

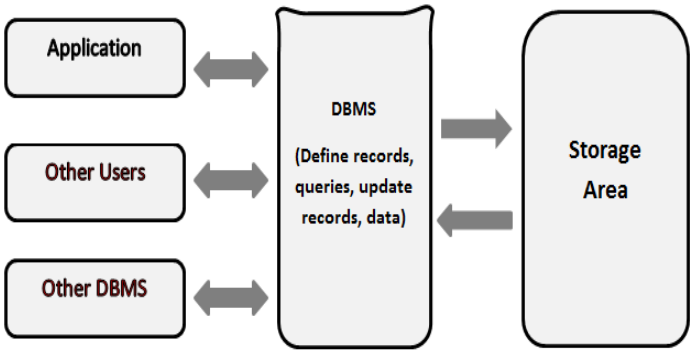


Figure 46: database algorithm

5.1.3 Types Of DataBase

Hierarchical

This type of DBMS employs the "parent-child" relationship of storing data. This type of DBMS is rarely used nowadays. Its structure is like a tree with nodes representing records and branches representing fields.

Network DBMS

This type of DBMS supports many-to-many relations. This usually results in complex database structures.

Relational DBMS

this type of DBMS defines database relationships in the form of tables, also known as relations. Unlike network DBMS, RDBMS does not support many to many relationships. Relational DBMS usually have pre-defined data types that they can support. This is the most popular DBMS type in the market.

Object-Oriented DBMS

this type supports the storage of new data types. The data to be stored is in the form of objects. The objects to be stored in the database have attributes (i.e. gender, ager) and methods that define what to do with the data. PostgreSQL is an example of an object-oriented relational DBMS.

5.1.4 Advantages and Disadvantages Of Using Database

Advantages	**Disadvantages**
Reduced data redundancy.	Complexity
Reduced updating errors and increased consistency.	Cost
Improved data access to users through the use of host and query languages.	Security
Reduced data entry, storage, and retrieval costs	Compatibility

5.1.5 Database Management System

A database management system (DBMS) is a software package designed to define, manipulate, retrieve and manage data in a database. A DBMS generally manipulates the data itself, the data format, field names, record structure, and file structure. It also defines rules to validate and manipulate this data.

5.1.6 Relational DBMS

All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL, and Microsoft Access are based on RDBMS.

It is called Relational Database Management System (RDBMS) because it is based on the relational model introduced by E.F. Codd, Standard relational databases enable users to manage predefined data relationships across multiple databases.

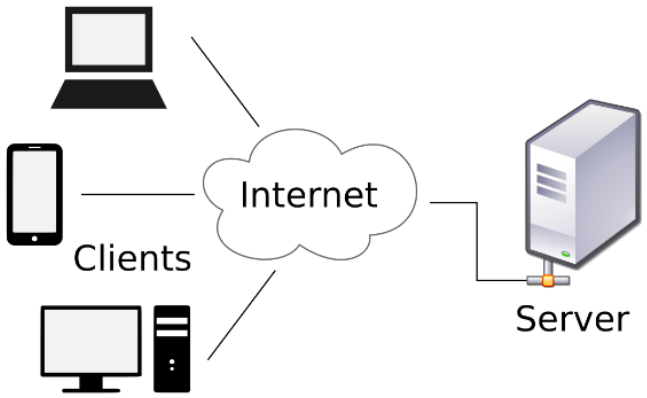


Figure 47: relations DBMS

5.1.7 MySQL Database

MySQL Definition

MySQL is a relational database management system based on SQL. The application is used for a wide range of purposes, including data warehousing, e-commerce, and logging applications. The most common use for MySQL however, is for the purpose of a web database.

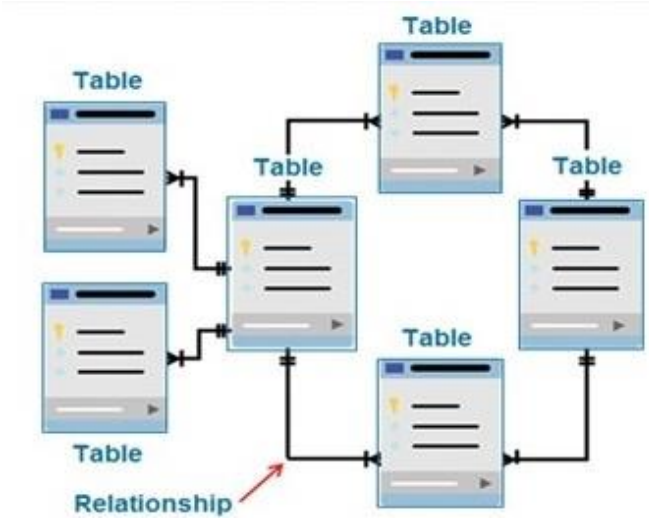


Figure 48: MySQL structure.

Advantages Of Using MySQL

1. MySQL is a database management system.

   A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server.
2. MySQL databases are relational.

   A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed
3. MySQL software is Open Source.

   Open Source means that it is possible for anyone to use and modify the software
4. MySQL Server works in client/server or embedded systems.

   The MySQL Database Software is a client/server system that consists of a multithreaded SQL server
5. The MySQL Database Server is very fast, reliable, scalable, and easy to use.
6. A large amount of contributed MySQL software is available.

MYSQL data types

Elements stored within database can be of type:

DATE TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 214748-3647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LONGBLOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (-8388608 to 8388607)	BOOLEAN	TINYINT(1)

Figure 49: MYSQL datatypes

MYSQL Python Connector

MySQL Connector/Python enables Python programs to access MySQL databases, using an API that is compliant with the Python Database API Specification v2.0 (PEP 249).

phpMyAdmin is a free software tool written in PHP that is intended to handle the administration of a MySQL or MariaDB database server. we could **use phpMyAdmin** to perform most administration tasks, including creating a database, running queries, and adding user accounts. But we needed the interaction with database to be in python to be connected properly with model and GUI. That's why this library is essential and preferred over phpMyAdmin interface in our case.

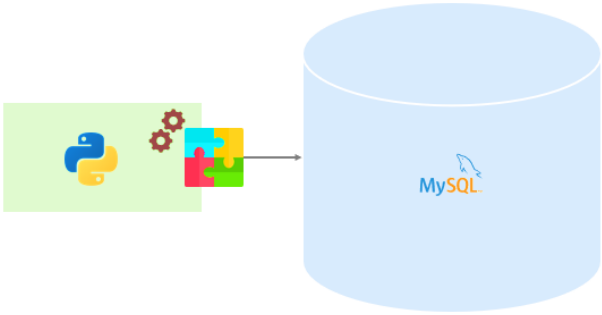


Figure 50: python MySQL connector

Create Our Database and Table

After setting mysql server on raspberry pi, a user is created @localhost with a secure password and given all high access privileges on raspberry pi terminal, create and modify.

First, we created our database on terminal to be accessed on python: `CREATE DATABASE dd;`

Then we use this user to establish a connection on python to the server and select the created database:

```
db=mysql.connector.connect(host="localhost",user="mohamed",password="12345",
db="dd")
```

Then we create an object "cursor" and use it to execute all queries on database:

```
c=db.cursor()
```

Table 7: database distraction table.

date	time	counter	Type1	Type2	Type3	Type4	Type5	Type6	Type7	Type8	Type9	Type 10

We created a table to contain:

- Date of distraction (column 1) datatype: `DATE`.
- Time of distraction (column 2) datatype: `TIME`.
- Counter that increases by one whenever a new distraction happens(column3) datatype: `INT`.
- Counter for each type of distraction (columns 4-13) datatype: `INT`.

We use timedate library to get date and time of distraction occurring.

date is a timedate object which is stored in tabe in string format: *"Year-Month-Day"* as string.

Time is also timedate object which is stored in string format*:"Hour:Minute:Second"*

All counters are integer data types calculated after processing on each frame.

Storing in database

The basic idea is store date in string type variables in python then use the cursor object to perform INSERT INTO `(table_name)` query.

1. The data must be stored in 13 variables.
2. All variables are put in 1 tuble.
3. Execute INSERT`()` query by the cursor object to put the tuble as a row in the table
4. Use cursor to commit data.

Table 8: database distraction table after filling data.

date	time	Count	Type1	Type2	Type3	Type4	Type5	Type6	Type7	Type8	Type9	Type 10
2021-07-06	10:05:02	1	0	0	0	0	0	1	0	0	0	0

So, this how one row is inserted with each element put in each column, one row will be inserted on every frame captured, the row will contain data processed on that frame.

Fetch From Database

We needed to fetch stored data from database to show it in GUI part,To fetch from database the `SELECT` query is used, and use `fetchall.()` function to put output in a list.

So, if we want to select all elements in table:

```
c.execute("SELECT * FROM table_name")
```

```
myresult=c.fetchall()
```

`myresult` is a `list` containing `tubles`, in which every `tuble` refer to data in a row in table:

`[(,),(,)]`.

But when we fetch some columns to create plots column name is typed instead of *:

```
c.execute("SELECT column_name FROM table_name")
```

```
myresult=c.fetchall()
```

When we fetch from columns at specific values, like fetching distraction types happening at day where date=’2021-07-07’

So, we specify the requested output and the

```
c.execute("SELECT * FROM distraction WHERE tdate='2021-6-28'")

 myresult=c.fetchall()
```

date	time	Count	Type1	Type2	Type3	Type4	Type5	Type6	Type7	Type8	Type9	Type10
2021-07-06	10:05:02	1	0	0	0	0	0	1	0	0	0	0
2021-07-07	2:15:22	2	0	0	0	0	0	1	0	0	1	0

so all other data on that date (row 3) is collected and in list type inside a tuble.

5.2 Graphical User Interface (GUI) Definition

The graphical user interface is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicator such as primary notation, instead of text-based user interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLIs), which require commands to be typed on a computer keyboard.

5.3 Best Python GUI Frameworks

Python has loads of frameworks for developing GUIs, the most popular Python GUI frameworks are listed below:

1. PyQt5
2. Kivy
3. WxPyThon
4. Libavg
5. PyForms
6. Tkinter – we used this framework to build out GUI.

5.4 Tkinter Framework

Tkinter is an open source, portable graphical user interface (GUI) library designed for use in Python scripts. Tkinter relies on the Tk library, the GUI library used by Tcl/Tk and Perl, which is in turn implemented in C. Therefore, Tkinter can be said to be implemented using multiple layers.

5.4.1 Tkinter Modules

Most of the time, `tkinter` is all you really need, but a number of additional modules are available as well. The Tk interface is located in a binary module named `tkinter`. This module contains the low-level interface to Tk, and should never be used directly by application programmers. It is usually a shared library (or DLL), but might in some cases be statically linked with the Python interpreter.

In addition to the Tk interface module, `tkinter` includes a number of Python modules:

- `tkinter.constants` – being one of the most important.
- `tkinter .colorchooser` – Dialog to let the user choose a color.
- `tkinter .filedialog` – Base class for the dialogs defined in the other modules listed here.
- `tkinter .filedialog` – Common dialogs to allow the user to specify a file to open or save.
- `tkinter .font` – Utilities to help work with fonts.
- `tkinter .messagebox` – Access to standard Tk dialog boxes.
- `tkinter .scrolledtext` – Text widget with a vertical scroll bar built in.
- `tkinter .simpledialog` – Basic dialogs and convenience functions.
- `tkinter .dnd` – Drag-and-drop support for tkinter. This is experimental and should become deprecated when it is replaced with the Tk DND.
- `turtle` – Turtle graphics in a Tk window.

5.4.2 Advantages of Tkinter

Layered approach

The layered approach used in designing Tkinter gives Tkinter all of the advantages of the TK library. Therefore, at the time of creation, Tkinter inherited from the benefits of a GUI toolkit that had been given time to mature. This makes early versions of Tkinter a lot more stable and reliable than if it had been

rewritten from scratch. Moreover, the conversion from Tcl/Tk to Tkinter is really trivial, so that Tk programmers can learn to use Tkinter very easily.

Accessibility

Learning Tkinter is very intuitive, and therefore quick and painless. The Tkinter implementation hides the detailed and complicated calls in simple, intuitive methods. This is a continuation of the Python way of thinking, since the language excels at quickly building prototypes. It is therefore expected that its preferred GUI library be implemented using the same approach.

Portability

Python scripts that use Tkinter do not require modifications to be ported from one platform to the other. Tkinter is available for any platform that Python is implemented for, namely Microsoft Windows, X Windows, and Macintosh. This gives it a great advantage over most competing libraries, which are often restricted to one or two platforms. Moreover, Tkinter will provide the native look-and-feel of the specific platform it runs on.

Availability

Tkinter is now included in any Python distribution. Therefore, no supplementary modules are required in order to run scripts using Tkinter.

5.4.3 Disavantages of Tkinter

The multi-layered approach taken in designing Tkinter can have some disadvantages as far as execution speed is concerned. While this could constitute a problem with older, slower machines, most modern computers are fast enough so as to cope with the extra processing in a reasonable time. When speed is critical, proper care must be taken so as to write code that is as efficient as possible.

5.4.4 Tkinter Widgets

Tkinter widgets are a subset of classes in Tkinter. They all inherit from the class Widget. Each one of them, when instantiated, creates a window component that can be styled to respond to the programmer's need. To control the appearance of a widget, you usually use options rather than method calls. Typical options include text and color, size, command callbacks, etc. (see table below) When widgets are instantiated, they don't immediately appear on the screen. Geometric managers (pack, grid, place) have to be called in order to make the widget visible on the screen. These last are also classes which inherit from the widget class. Aside from the methods inherited from the parent class Widget, each one of them has distinct options and

methods suitable for configuration of the particular widget. Therefore, Tkinter widgets don't only provide a tool to build a graphical interface, it also allows the programmers to create a fully functional application.

Common Widget Options

Table 9: common widget options.

Option	Value	Effect
Foreground (fg)	Colour	Changes the foreground colour. (Colour may be specified using pre-defined keywords or using RGB values)
Background (bg)	Colour	Changes the background colour. (Colour may be specified using pre-defined keywords or using RGB values)
Bd (border width)	Integer	Specifies the width of the widget border
Command	Callbacktype	Specifies which method is to be executed when the widget is selected.
Font	Font type	Specifies the font used by the widget. Often represented in a tuple (family, size, weight)
Padx, pady	Integer	Specifies the width between the current widget and the neighbor widgets.
Relief	Relief type: GROOVE SUNKEN RIDGE FLAT	Specifies the relief of the widget.
Text	String	Specifies the text appearing on the widget at run time

Common Widget Methods

To configure any widget's options, all widget implements the same configuration interface:

Table 10: common widget methods

Interface method	Return Value	Effect
widgetclass(master, option = value)	String	Creates an instance of this widget using the given options. (Options may be modified later).
cget(option)	String	Returns the current value of the specified option.
configure(option = value)	none	Configures option(s) for the widget.
keys( )	list	Returns a list of all options that can be set for this widget.

Types Of Widgets

Toplevel

This widget works pretty much like Frames, except it is displayed in a separate, top-level window. Toplevel usually have title bars, borders and other "window's decorations"

Table 11: the top-level widget options.

Options	width, height	Specifies the size of toplevel window
	cursor	Specifies the cursor to show when the cursor is within the toplevel window
	menu	A menu to be associated with this window

Frame

A Frame is rectangular region on the screen. The frame widget is mainly used as a geometry master for other widgets, or to provide padding between other widgets.

Table 12: the frame widget options.

Options	width, height	Specifiesthe size of toplevel window.
	cursor	Specifies the cursor to show when the cursor is within the frame.
	takefocus	Indicates that the user can use the Tab key to move to this widget.

Label

The Label widget is a standard Tkinter widget used to display a text or image on the screen. The button can only display text in a single font, but the text may span more than one line. In addition, one of the characters can be underlined, for example to mark a keyboard shortcut.\

Table 13: The Label widget options.

Options	text	The text to be displayed by the label.
	textvariable	Associates a Tkinter variable (usually a StringVar) to the label. If the variable is changed, the label text is updated.
	justify	Defines how to align multiple lines of text. Use LEFT, RIGHT, or CENTER.
	bitmap	The bitmap to display in the widget. If the image option is given, this option is ignored.
	image	The image to display in the widget. If specified, this takes precedence over the text and bitmap options.

Button

The Button widget is a standard Tkinter widget used to implement various kinds of buttons. Buttons can contain text or images, and you can associate a Python function or method with each button. When the button is pressed, Tkinter automatically calls that function or method.

Table 14: the button widget options and methods.

Options	anchor	Controls where in the button the text (or image) should be located.
	command	A function or method that is called when the button is pressed. The callback can be a function, bound method, or any other callable Python object.
	state	The button state: Normal, Active or Disabled.
	text	The text to be displayed in the button. The text can contain newlines. If the bitmap or image options are used, this option is ignored.

Entry

The Entry widget is a standard Tkinter widget used to enter or display a single line of text.

Table 15: the entry widget options and methods.

Options	show	Controls how to display the contents of the widget. If non-empty, the widget displays a string of characters instead of the actual contents. To get a password entry widget, use "*"
	state	The Entry state: Normal or Disabled

Notebook

The Notebook widget manages a collection of windows and displays a single one at a time. Each child window is associated with a tab, which the user may select to change the currently-displayed window.

Options	height	If present and greater than zero, specifies the desired height of the pane area (not including internal padding or tabs). Otherwise, the maximum height of all panes is used.
	padding	Specifies the amount of extra space to add around the outside of the notebook. The padding is a list up to four length specifications left top right bottom. If fewer than four elements are specified, bottom defaults to top, right defaults to left, and top defaults to left.
	width	If present and greater than zero, specified the desired width of the pane area (not including internal padding). Otherwise, the maximum width of all panes is used.
Tab options	state	Either “normal”, “disabled” or “hidden”. If “disabled”, then the tab is not selectable. If “hidden”, then the tab is not shown.
	sticky	Specifies how the child window is positioned within the pane area. Value is a string containing zero or more of the characters “n”, “s”, “e” or “w”. Each letter refers to a side (north, south, east or west) that the child window will stick to, as per the grid() geometry manager.
	padding	Specifies the amount of extra space to add between the notebook and this pane. Syntax is the same as for the option padding used by this widget.

	text	Specifies a text to be displayed in the tab.
	image	Specifies an image to display in the tab. See the option image described in Widget.
	underline	Specifies the index (0-based) of a character to underline in the text string. The underlined character is used for mnemonic activation if Notebook.enable_traversal() is called.

Treeview

The ttk.Treeview widget displays a hierarchical collection of items. Each item has a textual label, an optional image, and an optional list of data values. The data values are displayed in successive columns after the tree label. The order in which data values are displayed may be controlled by setting the widget option displaycolumns. The tree widget can also display column headings. Columns may be accessed by number or symbolic names listed in the widget option columns.

Each item is identified by a unique name. The widget will generate item IDs if they are not supplied by the caller. There is a distinguished root item, named {}. The root item itself is not displayed; its children appear at the top level of the hierarchy. Each item also has a list of tags, which can be used to associate event bindings with individual items and control the appearance of the item.The Treeview widget supports horizontal and vertical scrolling, according to the options described in Scrollable Widget Options and the met ods Treeview.xview() and Treeview.yview().

This widget accepts the following specific options:

Option	Description
columns	A list of column identifiers, specifying the number of columns and their names.
displaycolumns	A list of column identifiers (either symbolic or integer indices) specifying which data columns are displayed and the order in which they appear, or the string “#all”.
height	Specifies the number of rows which should be visible. Note: the requested width is determined from the sum of the column widths.

padding	Specifies the internal padding for the widget. The padding is a list of up to four length specifications.
show	A list containing zero or more of the following values, specifying which elements of the tree to display. tree: display tree labels in column #0. headings: display the heading row. The default is "tree headings", i.e., show all elements. Note: Column #0 always refers to the tree column, even if show="tree" is not specified.

Column Identifiers

- column(*column*, *option=None*, ***kw*)

Query or modify the options for the specified *column*. If *kw* is not given, returns a dict of the column option values. If *option* is specified then the value for that *option* is returned. Otherwise, sets the options to the corresponding values.

- heading(column, option=None, **kw)

Query or modify the heading options for the specified *column*. If *kw* is not given, returns a dict of the heading option values. If *option* is specified then the value for that *option* is returned. Otherwise, sets the options to the corresponding values.

- insert(*parent*, *index*, *iid=None*, ***kw*)

Creates a new item and returns the item identifier of the newly created item. Parent is the item ID of the parent item, or the empty string to create a new top-level item. *index* is an integer, or the value "end", specifying where in the list of parent's children to insert the new item. If *index* is less than or equal to zero, the new node is inserted at the beginning; if *index* is greater than or equal to the current number of children, it is inserted at the end. If *iid* is specified, it is used as the item identifier; *iid* must not already exist in the tree. Otherwise, a new unique identifier is generated.

Screen Layout

Fonts

As mentioned earlier, font in widgets is defined through the font option in the standard Tkinter widget interface. To properly define a font for widgets, this has to be specified using a font descriptor which is a n-tuple: (family, size, option1, option2…). The first element, family, specifies the font name. Tkinter supports a vast variety of font names on different platform such as Arial, Times New Roman, etc. It also recognizes specific system fonts like 6x10. The second element of the tuple is the size, specified via an integer. The options that follows the size parameter within the tuple are used to describe the style of the font. This last can be normal, bold, underlined, italic, etc.

Colours

Widget's colour display can be configured via the colour option. The option can be specified using a common colour name such as red, blue, green, etc. However, Tkinter includes a color database which maps color names to the corresponding RGB values, therefore, it also recognizes more exotic names such as mocassin, peachpuff, etc.

If desired, the user can enter his or her own colour using RGB format. The format is #RRGGBB, which is then capable of having 65536 different kind of colours.

Steps for create any widget

1. Load a widget class from Tkinter
2. Make an instance of it (repeat 1 and 2 as needed)
3. Arrange the widget in its parent widget
4. Enter the event loop

5.4.5 Tkinter Variables

Tkinter provides 4 descendants of the Variable class, each implementing its specific get( ) method:

- StringVar: used for string values
- IntVar: used for integer values
- DoubleVar: used for double precision floating point values
- BooleanVar: used for Boolean values (0 or 1)

5.4.6 Geometry management in Tkinter

Geometry management consists of placing widget placement and size on the screen. Tkinter provides three geometry managers, therefore allowing the user to quickly and efficiently build user interfaces, with maximal control over the disposition of the widgets. The geometry managers also allow the user make abstraction of the specific implementation of the GUI on each platform that Tkinter supports, thus making the design truly portable.

Geometry management implies a great deal of negotiations between the different widgets and their containers, as well as the windows they are placed in. Geometry management is not only responsible for the precise position and size of a widget, but also of this widget's position and size relative to the other widgets, and of renegotiating these factors when conditions change, for example when the window is resized.

5.4.7 The Pack geometry manager

The packer is the quickest way to design user interfaces using Tkinter. It allows the user the place the widgets relative to their contained widget. It is the most commonly used geometry manager since it allows a fair amount of flexibility.

The packer positions the slave widgets on the master widget (container) from the edges to the center, each time using the space left in the master widget by previous packing operations (see demo).

Options for the Pack geometry manager:

Option	Values	Effect
expand	YES (1) NO (0)	Specifies whether the widget should expand to fill the available space (at packing time and on window resize)
fill	NONE X Y BOTH	Specifies how the slave should be resized to fill the available space (if the slave is smaller than the available space)
side	TOP(default) BOTTOM RIGHT LEFT	Specifies which side of the master should be used for the slave.

anchor	N,S, W, E NW SW NE SE NS EW NSEW CENTER	Specifies where the widget should be placed in the space that it has been allocated by the packer, if this space is greater than the widget size. Default is CENTER.

Methods supported by the Pack geometry manager:

Method	Effect
pack(option=value,…) pack_configure(option=value,…)	Packs the widget with the specified options.

5.4.8 The Grid geometry manager

The grid geometry manager is used for more complex layouts. It allows the user to virtually divide the master widget into several rows and columns, which can then be used to place slave widgets more precisely. The packer geometry manager would require the use of multiple nested frames to obtain the same effect.

The grid geometry manager allows the user to build a widget grid using an approach that is very similar to building tables using HTML. The user builds the table specifying not only the row and column at which to place the widget, but also the rwo span and column span values to use for the widget. In addition to that, the sticky option can allow almost any placement of the widget inside a cell space that is bigger than the widget itself. Combinations of values for the sticky option also allow to resize the widget, such as EW value, equivalent to an expand option combined with a fill=X for the packer.

Options for the Grid geometry manager

Option	Values	Effect
row, column	Integer values	Specifies where the widget should be positioned in the master grid.

Methods supported by the Grid geometry manager:

Method	Effect
grid(option=value,…), grid_configure(option=value,…)	Places the widget in a grid, using the specified options.
grid_location(x,y)	Returns a tuple (column, row) which represents the cell in the grid that is closest to the point (x, y).

5.4.9 The Place geometry manager

The place geometry manager is the more powerful manager of all since it allows exact placement of the widgets inside a master widget (container). However, it is more difficult to use and usually represent a great amount of overhead that's is generally not needed. The Place geometry manager allows placement of widgets using either exact coordinates (with the x and y options), or as a percentage relative to the size of the master window (expressed as a float in the range [0.0, 1.0]) with the relx and rely options. The same principle holds for the widget size (using width / height and/or relwidth / relheight).

The options supported by the place geometry manager are as follows:

Option	Type	Effect
Anchor	String, one of: N NE E SE SW W NW (Default) CENTER	Specifies which part of the widget should be placed at the specfied position.
Bordermode	INSIDE, OUTSIDE	Specifies if the outside border should be taken into consideration when placing the widget.

in (in_)	Widget	Places the slave in the master passed as the value for this option.
relwidth, rel-height	Float [0.0, 1.0]	Size of the slave widget, relative to the size of the master.
relx, rely	Float [0.0, 1.0]	Relative position of the slave widget.
width, height	integer	Absolute width and height of the slave widget.
x, y	Integer	Absolute position of the slave widget.

5.5 Driver Distraction Detection GUI

All these pictures were taken from LCD screen of our device.

5.5.1 The Login window

consists of

1. Two label widgets for displaying "user name", "password".
2. Two entry widgets for entering the driver's name and password.
3. A button widget to log in and go to the start window.
4. A background image.

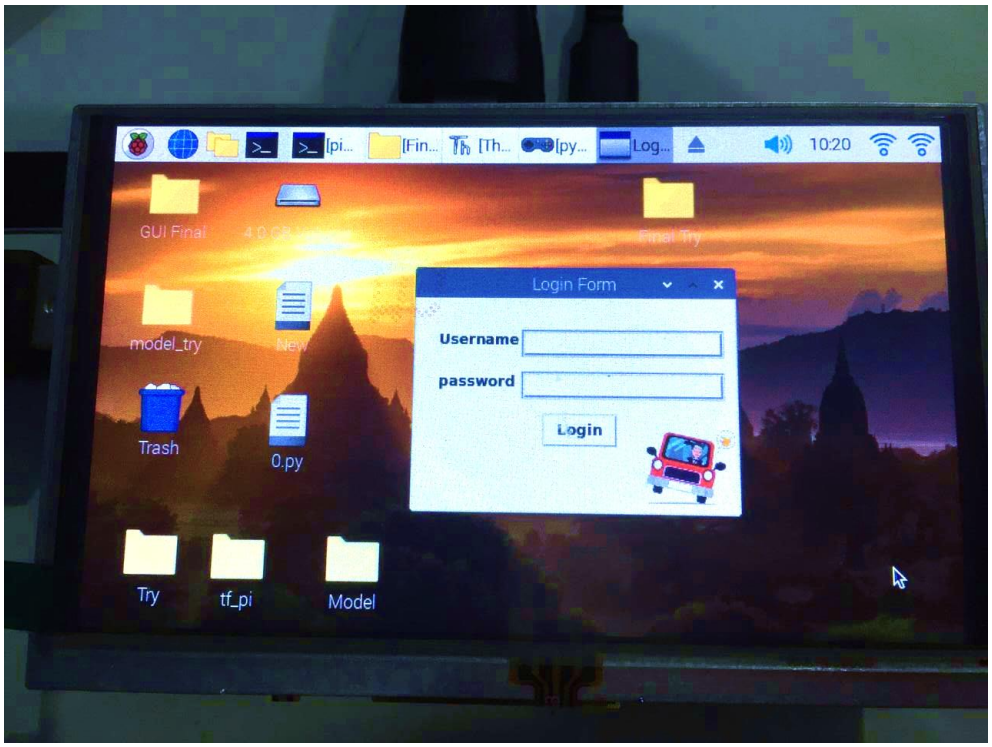


Figure 51: login window.

5.5.2 The Start Widget

Consists of

- A '**Start**' button for start capturing the frames and send it to the model for detection.
- A '**Show**' button one to display the previous distractions statistics.
- A label widget contains an image to appear in the background.

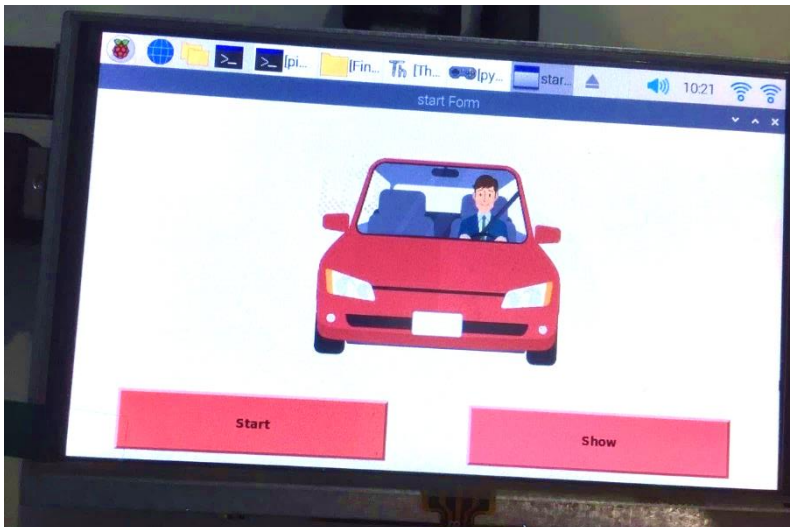


Figure 52: the start window

5.5.3 The Driving Window

Consists of:

- A '**Pause**' widget button moves to the "**Pause Window**", in addition, to stop capturing frames until the driver presses the '**Continue**' button which will appear in the neat window.
- A '**Stop**' stop capturing frames and display the distractions statistics during the trip as well as the previous graphs.

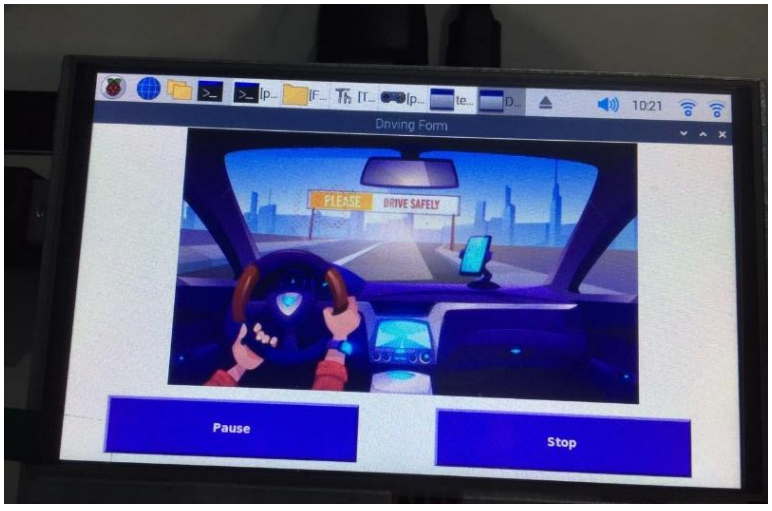


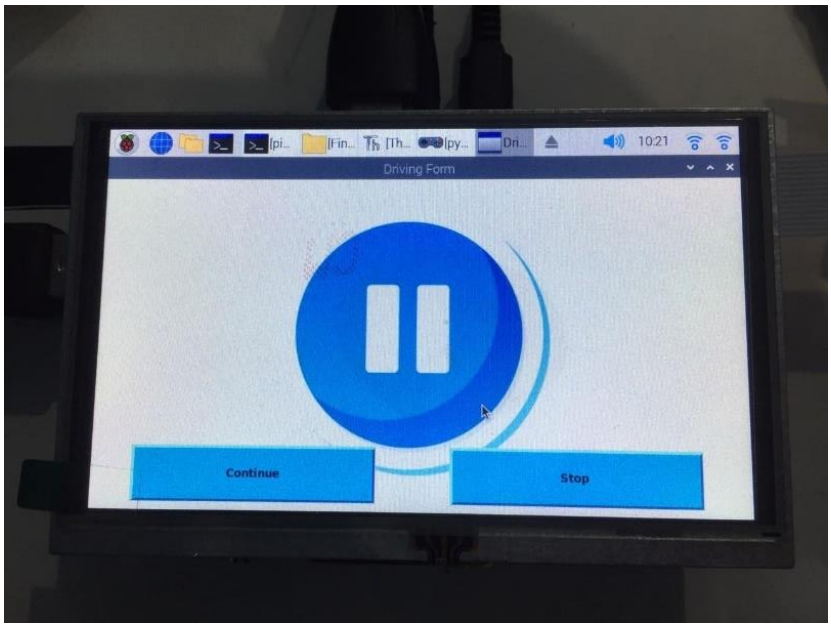
Figure 53: the driving window.

- A label widget contains an image to appear in the background.

5.5.4 The Pause Window

Consists of:

- A 'Continue' widget button to continue capturing the frames during driving and moves to "Driving Window".
- A 'Stop'
- stop capturing frames and display the distractions statistics during the trip as well as the previous graphs.



- A label widget contains an image to appear in the background.

Figure 54: the pause window.

5.5.5 The Statistics Window

- Using the notebook widget, matplotlip python library, and FigureCanvasTkAgg , we create a three taps as shown:
  1. **A distraction numbers per day tap -** showing a graph between the date of the current and previous days with the number of distracted actions that happened. these data can be returned from our database.
  2. **An abstract distraction table tap –** showing the type of distraction and and how often it occurs on this day using treeview widget.

3. **A Pie graph** – showing driver distraction rate during the trip.

- A '**Go to start**' button move to the "**Start Window**" and initialization a new trip, in addition to, delete all the saved frames during this trip.
- A '**Exit**' button terminates the application.

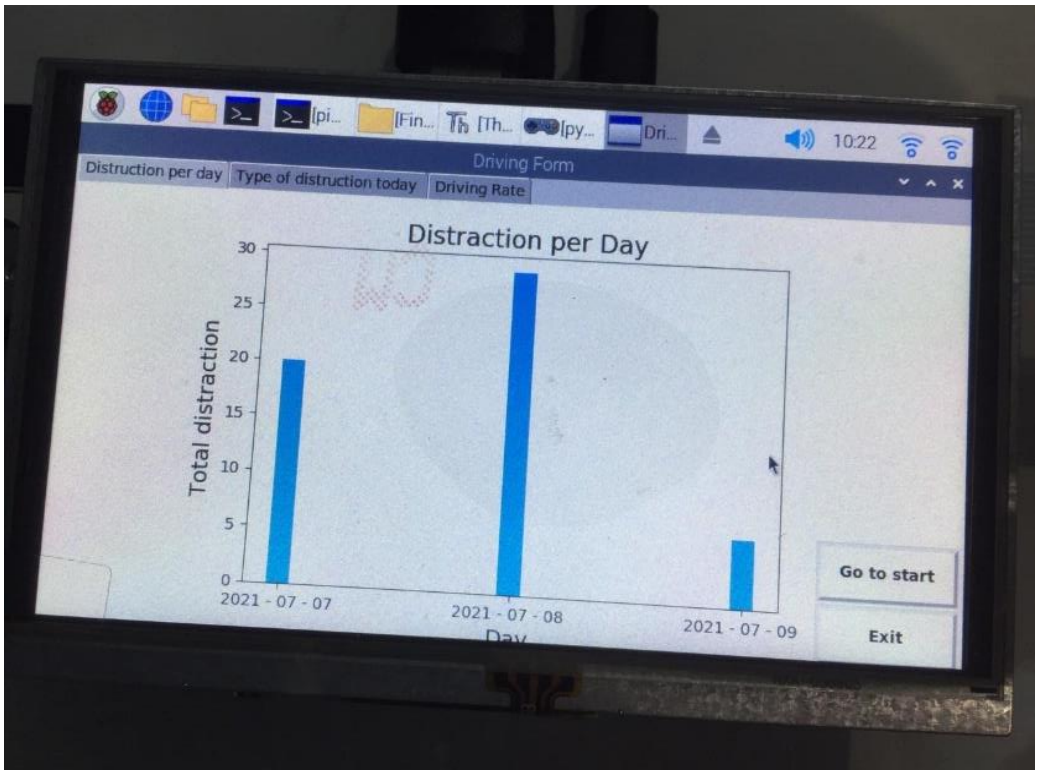


Figure 55: tap one - number of distractions per day.

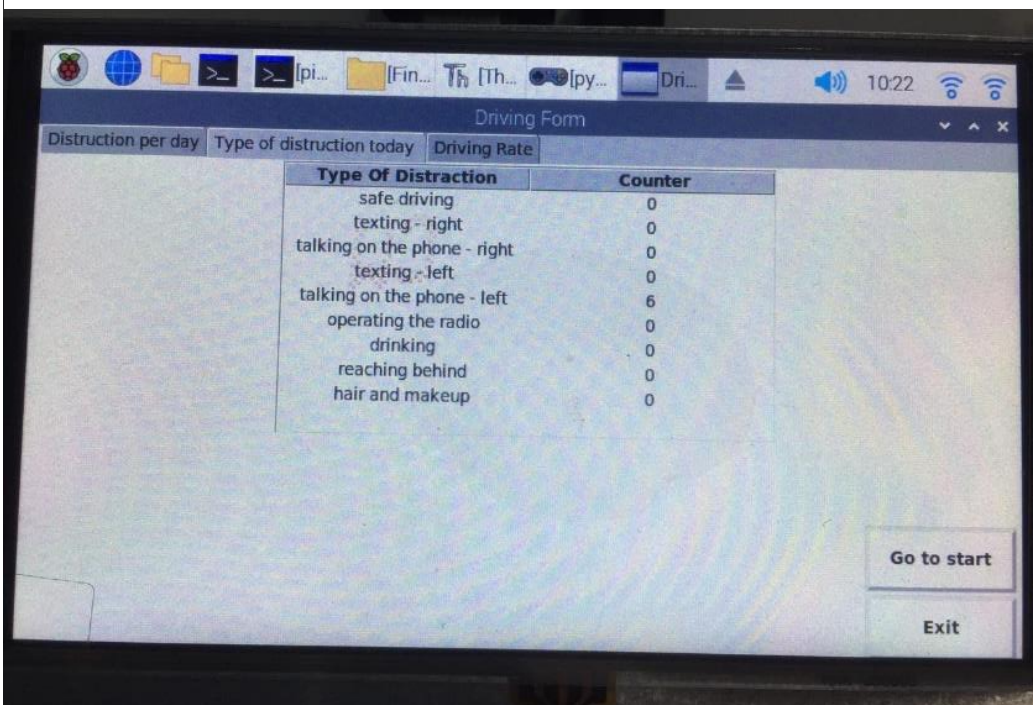


Figure 56: tap one - the abstract distraction table tap.

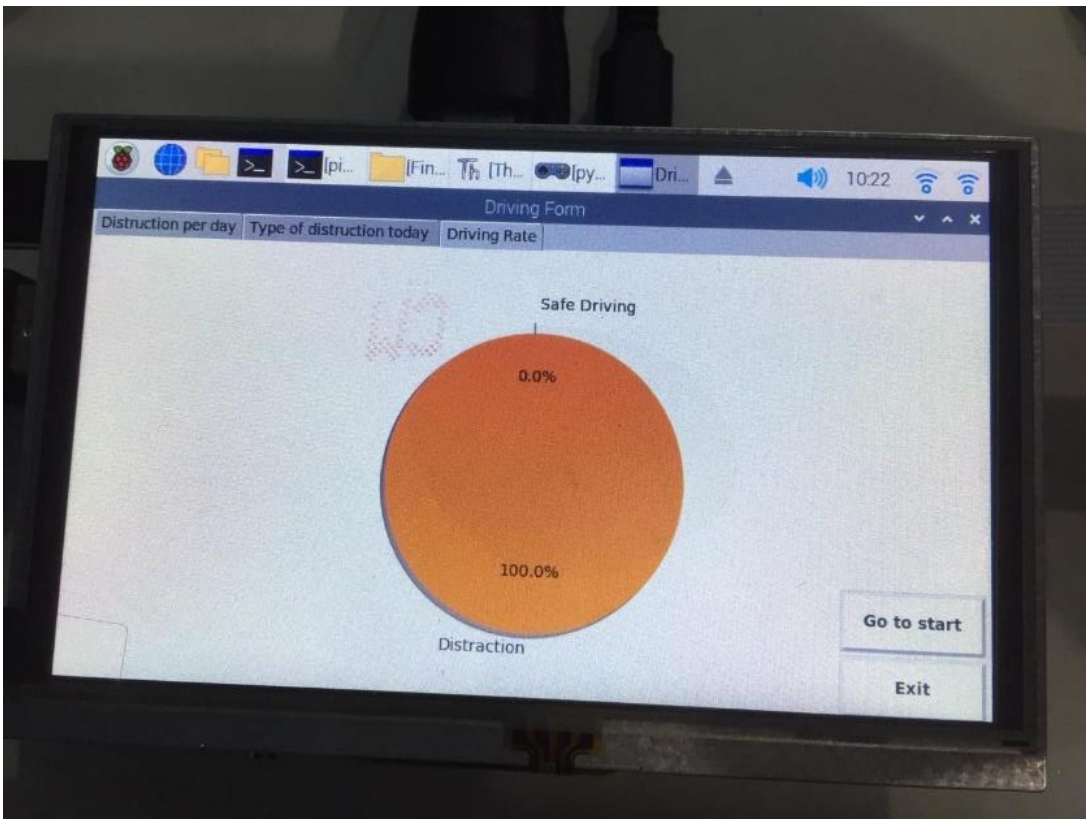


Figure 55: tap three - pie graph for distraction rate per day.

CONCLUSION AND FUTURE WORK

Our project presents a method for driver distraction detection and evaluation while performing a secondary task. The detection is executed by the deep learning algorithm based on the CNN architecture. Therefore, the method is capable not only to detect driver behavior, but also alert the driver and using a touch screen to display data regarding the driver's performance while driving on the current day, as well as those taken in the previous days. The driver can also control the device by temporarily or permanently turning it on or off.

As an extension of this work, we are working towards lowering the number of parameters and computation time. Incorporating temporal context may help in reducing misclassification errors and thereby increasing the accuracy. Also, in future, we wish to develop a system that will detect visual and cognitive distractions as well along with manual distractions. It will also be easier for the driver if the device is linked to a mobile application. Adding extra sensors to the device to maintain the driver's safety, as well as linking the outputs of these sensors to the driver's phone so that, in the case of a medical emergency or an accident, a call may be made to the emergency numbers on the list.

In the future, this technology could be applied in smart cities to detect driver distraction automatically and then send a warning message to the driver to prevent accidents. This technique can allow law-enforcement authorities to identify a distracted driver and monitor them using radar and cameras. Once detected, these drivers can be penalized. Moreover, recently developed semi-autonomous commercial cars require drivers to pay attention to road and traffic conditions. Autonomous steering-control systems require drivers to be ready to take control of the wheel. Thus, distracted-driver detection is an important system component in these cars.

Distraction detection can also be used to enable advanced driver-assistance system features, such as collision-avoidance systems that plan evasive maneuvers to reduce vehicle accidents and improve transportation safety, a system that can classify distracted driving is highly desirable and has attracted much research interest.

REFRANCES

[1] Geron, Aurélien, "*Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems"*, O'Reilly Media, 2019.

[2] El-amir, Hisham; Hamdy, Mahmoud., *"Deep learning pipeline: building a deep learning model with Tensor-Flow"*, Après, 2019.

[3] Quinn, J., McEachen, J., Fullan, M., Gardner, M., & Drummy, M. (2019). *"Dive into deep learning: Tools for engagement*", Corwin Press.

[4] Dewancker, Ian, Michael McCourt, and Scott Clark, *"Bayesian optimization for machine learning: A practical guidebook."*, arXiv preprint arXiv:1612.04858 (2016).

[5] Burkov, Andriy, *"The hundred-page machine learning book."*, Vol. 1. Canada: Andriy Burkov, 2019.

[6] Celona, Luigi, et al. "*A multi-task CNN framework for driver face monitoring."*, 2018 IEEE 8th International Conference on Consumer Electronics-Berlin (ICCE-Berlin). IEEE, 2018.

[7] Alberto Fernández , Rubén Usamentiaga , Juan Luis Carús, and Rubén Casado , *"Driver distraction using visual-based sensors and algorithms."*, Sensors 16.11 , 14 July 2016.

[8] Olson, R. L., Hanowski, R. J., Hickman, J. S., & Bocanegra, J.,*"Driver distraction in commercial vehicle operations. No. FMCSA-RRT-09-042. United States. Department of Transportation"*, Federal Motor Carrier Safety Administration, 2009.

[9] *Zhang, C., Li, R., Kim, W., Yoon, D., & Patras, P, "Driver behavior recognition via interwoven deep convolutional neural nets with multi-stream inputs."* IEEE Access 8 (2020): 191138-191151.

[10] Baheti, Bhakti, Suhas Gajre, and Sanjay Talbar. *"Detection of distracted driver using convolutional neural network."*, Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2018.

[11] *Sumit Saha, 15 Dec 2018 ,"A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way", Towards Data Science, 10 July 2021* <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

[12] Jason Brownlee on April 22, 2019, *"A Gentle Introduction to Pooling Layers for Convolutional Neural Networks*", machine learning mastery , *10 July 2021* <https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/ >.

[13] Anirudh VK 2019, *"Is The New Raspberry Pi 4 An Affordable Option For ML At The Edge"*, Analytics India Magazine, *7 July 2021* <https://analyticsindiamag.com/is-the-new-raspberry-pi-4-an-affordable-option-for-ml-at-the-edge/ >.

[14] Raspberry Pi Foundation, *"Getting started with Raspberry Pi", 10 July 2021* < https://projects.raspberrypi.org/en/projects/raspberry-pi-getting-started >.

[15] Onkar Sharma 11 October 2019, *"Introduction to Databases"*, C-sharp Conrner< https://www.c-sharpcorner.com/article/introduction-to-databases/ >.

[16] MySQL Tutorial from the MySQL 5.7 Reference Manual. PDF file. 2021<https://downloads.mysql.com/docs/mysql-tutorial-excerpt-5.7-en.pdf.>

[17] NumPy's library guide < https://numpy.org/doc/stable/user/index.html >.

[18] TensorFlow's library guide < https://www.tensorflow.org/guide >.

[19] Keras' library guide < https://keras.io/guides/ >.

[20] Panda's library guide < https://pandas.pydata.org/pandas-docs/stable/user_guide/index.html>.

[21] Seaborn's library guide < https://seaborn.pydata.org/tutorial.html >.

[22] Pickle's library guide < https://docs.python.org/3/library/pickle.html >.

[23] MYSQL's library guide < https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>.

[24] OpenCV's library guide < https://docs.opencv.org/3.4/d8/dfe/classcv_1_1VideoCapture.html >.