# SPOTHOLE PROJECT

CNN-based Pothole-Detection for Avoidance Road Accident

Submitted by: Group G270

Members:
Aditya Vidiyala
T. Sai Krishna Nair
Kanduri Adithya
Ehsaas Nahata
Vakiti Nigama Reddy
Udit kandi

Under the Guidance of: Hari Babu Sir

Keshav Memorial Engineering College

Date of Submission: 1st February 2025

# Acknowledgement

We would like to express our sincere gratitude to everyone who contributed to the successful development of this pothole detection system.First and foremost, we extend our heartfelt appreciation to our mentor Hari Babu Sir, whose guidance and technical expertise played a crucial role in shaping this project. His invaluable insights into machine learning, web development, and deployment helped us refine our approach and overcome challenges.

We are also grateful to our team members, whose dedication, collaboration, and continuous effort made this project a reality. Their contributions in coding, model training, data preprocessing, and UI/UX development were instrumental in ensuring the system's efficiency and usability.

A special thanks to open-source communities and researchers who provided publicly available datasets and resources on deep learning and computer vision, enabling us to develop a robust and scalable solution.

We would also like to acknowledge the support of our college Keshav Memorial Engineering College for providing the necessary resources, infrastructure, and encouragement to pursue this project.

Finally, we extend our gratitude to our family, friends, and peers for their constant motivation and feedback throughout the development process. Their encouragement helped us stay focused and driven to achieve our objectives.

This project would not have been possible without the collective efforts of all those mentioned above. We sincerely appreciate their support in bringing this innovative pothole detection system to life.

# Preface

Road safety is a critical concern in modern transportation systems, and potholes pose a significant threat to vehicles, passengers, and pedestrians. Timely detection and reporting of potholes can help authorities take necessary action, reducing accidents and vehicle damage. This project aims to leverage deep learning, computer vision, and web technologies to develop an efficient pothole detection system that automates the process of identifying and reporting road surface anomalies.

This document provides a comprehensive overview of the pothole detection system, covering all aspects of its development, including data collection, preprocessing, model selection, implementation, deployment, and scalability. It also highlights the system's key features, such as JWT-based authentication, data encryption for security, and a history section for users to track past detections.

The project integrates CNN object detection models, APIs and interfaces to ensure a seamless user experience. By implementing real-time detection and secure data handling, this system aims to potentially assist government agencies, transportation authorities, and the general public in maintaining road safety.

This preface serves as an introduction to the technical and functional aspects of the system, providing a clear understanding of its objectives, significance, and implementation strategies. We hope this project will contribute positively to road infrastructure monitoring and enhance public safety through the adoption of intelligent technologies.

# CNN-based Pothole-Detection for Avoidance Road Accident

Aditya Vidiyala
*Keshav Memorial Engineering College*
Hyderabad,Telangana
adityavidiyala0408@gmail.com

T. Sai Krishna Nair
*Keshav Memorial Engineering College*
Hyderabad,Telangana
saikrishnanair10@gmail.com

Kanduri Adithya
*Keshav Memorial Engineering College*
Hyderabad,Telangana
kanduriadithya16@gmail.com

Ehsaas Nahata
*Keshav Memorial Engineering College*
*Hyderabad,Telangana*
ehsaasnahata@gmail.com

Vakiti Nigama Reddy
*Keshav Memorial Engineering College*
*Hyderabad,Telangana*
nigamavakreddy@gmail.com

Udit kandi
*Keshav Memorial Engineering College*
*Hyderabad,Telangana*
uditkandi.ws@gmail.com

***Abstract*—Approaches for roadway pothole detection technology are being developed to provide offline data collection for road repair or real offline vehicle control (for vehicular applications or automated driving). The potential of pothole detecting technology to increase highway safety and lower the cost of road repair has made it a crucial research area. Researchers have been researching numerous methods for detecting potholes on roads, and one of the most promising is deep learning algorithms. Artificial neural networks are used by deep learning algorithms to extract patterns and characteristics from massive amounts of data. Several applications, including object identification and picture recognition, have successfully exploited this strategy. Pothole detection on road photos can be done using the same methods. Convolutional neural networks (CNN) are one such method. CNNs have been demonstrated to perform with excellent accuracy in a variety of image recognition applications because they are built to process images by learning relevant characteristics and patterns. Researchers have been investigating a number of methods for spotting potholes on roads all around the world due to these factors. Mass transit is supported by roads, which provide a significant economic contribution. In the transportation networks, potholes in the roads are a major source of worry. Many studies have recommended using deep learning algorithms, which deal with diverse image analysis and object detection methods, to automate pothole detection. The pothole detection method needs to be automated with the highest level of accuracy and dependability.**

# I INTRODUCTION

Both developed and developing countries struggle with the widespread issue of potholes, which has a negative impact on both the economy and society. Roads are the main form of transportation in many nations, and they can seriously hurt both the vehicles and the people inside. The second-largest road network in the world is found in India. As a result, the road system is crucial to the social and economic development of India. In contrast to the general annual GDP growth of 6%, the road transport sector GDP increased at an average yearly rate of close to 10% over the past ten years, according to the report . It had cost India between three and four percent of its GDP in 2019. According to the latest government data, between 2018 and 2020, 5,626 people died as a result of road accidents caused by potholes.The Indian Transportation and Highway Department has also stated that one of the main reasons for accidents is road deterioration. The Indian government currently builds roads fairly quickly. However, due to the inadequate drainage system and the overloaded vehicles.

Our model is able to identify potholes in the road and give drivers feedback in real-time This can assist drivers in slowing down or stopping their car before hitting a pothole, avoiding damage, and lowering the possibility of an accident. Implementing such a concept has significant financial advantages. For now our model is only able to identify if there is a pothole or not but we intend to modify our model in such a way that it can detect the pothole along with its exact location and give the real time data to the drivers.

Drivers shell out a sizable sum of money annually to fix their cars after hitting potholes.These repairs will cost more than Rs. 10,000 crores in India alone over a five- to seven-year period. In 2017, potholes claimed about 10 lives every day, totaling 3597 fatalities per year nationwide, a more than 50% increase from the toll for 2016 .This can greatly reduce these expenditures and raise general road safety by using a pothole detection model. In addition, infrastructure is crucial to economic growth because trade and transportation depend on well-maintained roads. By minimizing the damage induced by potholes, the system can ensure that our roads are in good shape, improving mobility and productivity. The economy will flourish from this, and society as a whole will improve.

In order to ensure that the road still meets the needs of services generally, human visual inspection is the primary way of assessing the condition of the road.

Manually examining and evaluating visual road data, however, is a time-consuming and expensive process. Also, the subjectivity and expertise of the human raters have a significant impact on the outcomes. Also, these techniques are exceedingly laborious and uncomfortable for inspectors. The development of science and technology, as well as the acceptance of the deep learning model in the engineering community, has made it possible to replace humans with sophisticated, intelligent systems that are inexpensive and low-cost. To address this issue, there is a need for an accurate model that can detect and alert drivers of potholes on the road. This can help drivers avoid potholes, which in turn can prevent damage to their vehicles and reduce the risk of accidents. Our pothole detection system uses deep learning and convolutional neural networks to accurately detect potholes on the road. It can help drivers avoid potholes and reduce the risk of damage to their vehicles. It can also help governments and municipalities identify areas where roads need to be repaired or maintained, which can ultimately lead to safer roads for everyone. So, to avoid this type of situation, an accurate model is to be created.

- To detect potholes.
- Stop or lower the speed of the vehicles
- To overcome issues related to potholes
- To save one’s capital.

Pothole detection requires a lot of manual labor and takes a lot of time. Many technologies, including vibration-based, 3D reconstruction-based, and vision-based methods, have been used to identify potholes. Yet each of these methods has its drawbacks. As a result, we have attempted to address the issue of potholes using a CNN.

This paper begins with a review of pothole detection methods, extending the brief annotations on related literature from those earlier conference papers.There are many datasets available for our project but we choose to make own dataset by collecting the images from the multiple datasets which we thought would be perfect for the model to train upon.

*A. Abbreviations and Acronyms*

CNN – Convolutional Neural Network

## II. RELATED WORK

Chen, Guo-Hong, et al.'s proposed deep learning method for pothole detection using convolutional neural networks (CNNs) is a significant breakthrough in the field of road maintenance and monitoring. The approach used a dataset of road images with potholes to train the CNN model. The model was designed to recognize the patterns and features associated with potholes from the input images. The study demonstrated that their CNN model was able to achieve high accuracy in detecting potholes from the images, making it a promising approach for effective road maintenance and monitoring.

In their review, Motwani, P., and Sharma, R. emphasized the significance of using deep learning techniques for pothole detection and classification. They also stressed the importance of data augmentation techniques to improve the performance of the models. Furthermore, they discussed various CNN architectures used for pothole detection, including LeNet, AlexNet, VGG, ResNet, and YOLO. The authors analyzed the strengths and weaknesses of each of these architectures and provided recommendations for choosing an appropriate model. They also discussed the limitations of current pothole detection models and suggested future research directions, such as incorporating additional data sources and exploring novel deep learning architectures. This review provides valuable insights into the state-of-the-art in pothole detection using deep learning techniques and can guide future research in this area.

Sharma, S. K., & Sharma, R. C. conducted a systematic literature review of pothole detection using deep learning. They discussed the limitations of existing approaches and identified potential areas for future research, such as developing more robust models for pothole detection in challenging weather conditions. This review provides a valuable resource for researchers looking to improve pothole detection using deep learning and can help accelerate progress in this field.

It has been demonstrated that in terms of pothole identification, CNN-based models outperform conventional techniques, including vibration-based, 3D reconstruction based, and vision-based techniques. This is because CNNs, as opposed to traditional approaches, which frequently require manual feature engineering and may not generalize well to new data, can automatically learn relevant features from raw data and improve their performance with more data.

Drawbacks of other techniques

- Methods based on vibration depend on spotting vibrations brought on by cars driving over potholes. However, their setup costs can be high, and they might not be effective in noisy or low-visibility environments .
- Methods based on 3D reconstruction recreate the road surface using 3D models, although the implementation of this strategy can be costly and time-consuming .
- Cameras are used in vision-based systems to find potholes; however, they can be hindered by bad illumination or obstructions

On the other hand, CNNs are effective at processing vast volumes of picture data and are useful for identifying intricate patterns like potholes.and are useful for identifying intricate patterns like potholes. They may be trained to find potholes in a variety of lighting and weather situations and have been found to achieve excellent accuracy in pothole detection tasks . Also, CNNs are more useful for real-world applications since they can be quickly installed on inexpensive devices like smartphones .

First, compared to vibration-based systems that necessitate specialist equipment, CNN-based models reliably detect potholes in real-time using only pictures or video data, making them less expensive and easier to set up . Also, unlike vision-based techniques, which have difficulty in such situations, CNN-based models are accurate even in dimly lit or foggy environments . Second, unlike 3D reconstruction-based techniques, which are constrained to detecting potholes of a given shape and size, CNN-based models can be trained to recognize potholes in a variety of sizes and shapes . Additionally, CNN-based models are quicker and simpler to implement than vision-based ones since they need fewer complicated algorithms. They can also be trained to become more accurate over time, making them more accurate and efficient than conventional techniques . Overall, compared to conventional methods, CNN-based models have shown improved pothole detection accuracy and reliability, making them a promising strategy for enhancing road safety.

## III. DEEP LEARNING MODEL FOR DETECTION PROCEDURE

Our suggested technique for detecting potholes is based on deep learning models, more especially convolutional neural networks (CNNs). A common deep learning model for image classification applications, such as object detection, is the CNN.

The model was trained using a binary cross-entropy loss function, and stochastic gradient descent (SGD) with momentum was utilized as the optimizer. Deep learning models known as convolutional neural networks (CNNs) are frequently employed in computer vision tasks like segmentation, object detection, and image categorization. Due to their capacity to collect both the spatial and temporal characteristics of images, CNNs are particularly well suited for tasks involving images. The Accuracy of our model came out to be 95 %. The intricate network topology of the model architecture enables it to record both low-level and high-level aspects of images. In particular, it combines Convolution layers, Pooling layers, and fully connected layers to extract characteristics from images at various degrees of abstraction. Hyperparameters are variables that affect how a neural network behaves, such as the number of layers and learning rate. Finding the ideal set of hyperparameters that produce the highest performance on the validation set is the goal of fine-tuning.

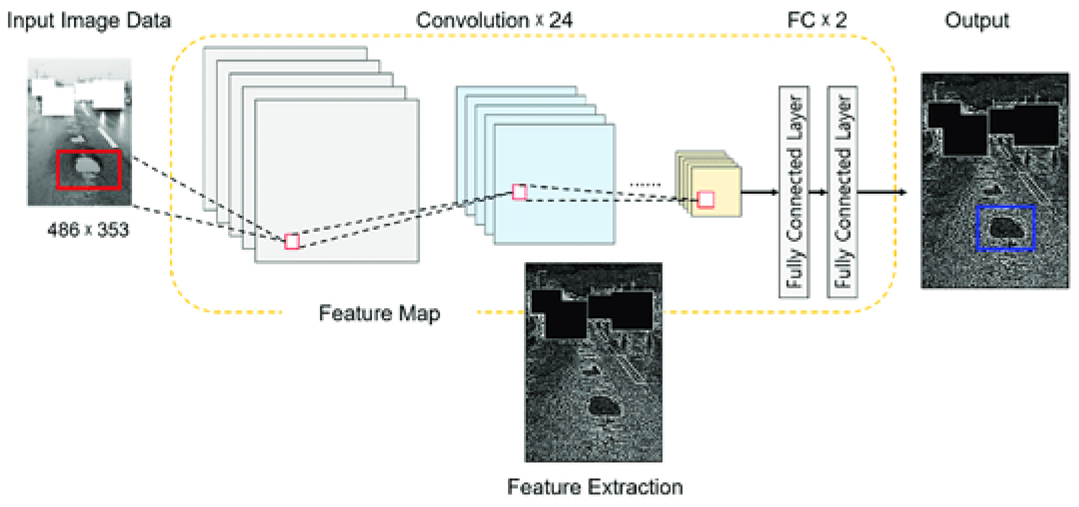


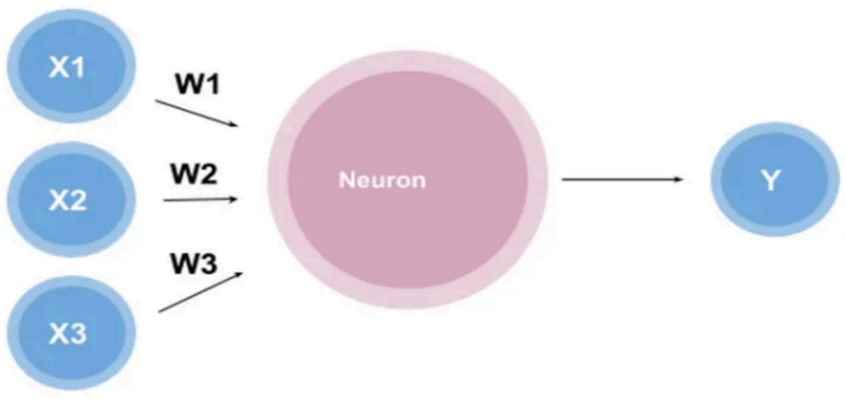
Fig.1 Block diagram of the project

# IV WHY USE CNN

## HOW THE CONCEPT OF NEURAL NETWORKS STARTED

### 1. PERCEPTRON

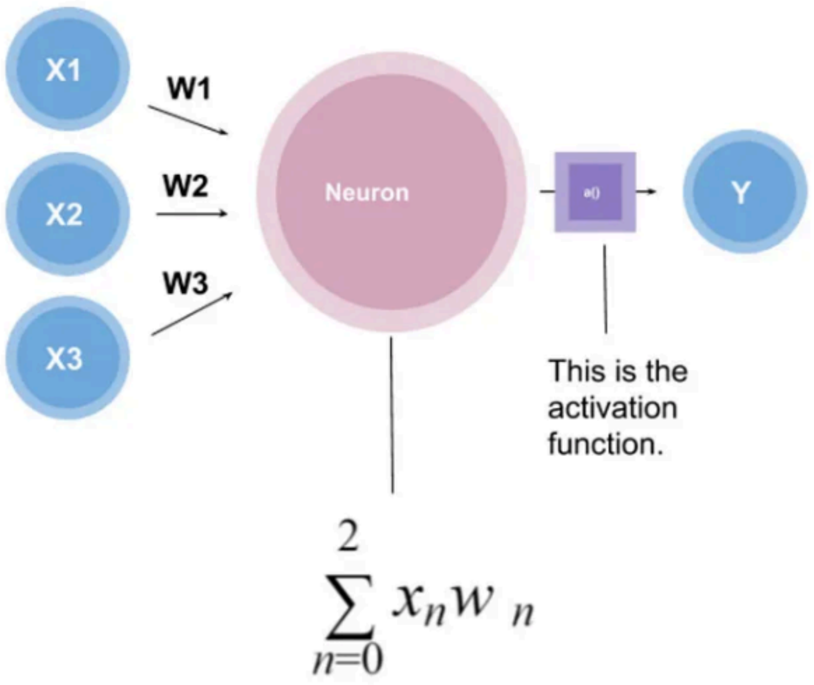
The Neural Networks started from the concept of *Perceptron*, But what is a perceptron and why is it used? How does it work?. A single layer perceptron is the basic unit of the neural network. A perceptron consists of input values, weights and bias, a weighted sum and activation function.The perceptron was first introduced by American psychologist, Frank Rosenblatt in 1957 at Cornell Aeronautical Laboratory. He was heavily inspired by the biological neuron and its ability to learn. A perceptron works by taking in some numerical inputs along with what is known as *weights* and a *bias.* It then multiples these inputs with the respective weights(this is called weighted sum). These products are then added together along with bias.The activation function takes the weighted sum and the bias as inputs and returns the final output.



$$y = x_1w_1 + x_2w_2 + x_3w_3$$

This function is called the weighted sum because it is the sum of the weights and inputs. This looks like a good function, but what if we wanted the outputs to fall into a certain range, say 0 to 1.

We can do this using something called an *Activation function*. An activation function is a function which converts the input given to it into a certain output based on the set of the rules. It also brings non-linearity to the data. There are many activation functions which are used according to the requirement



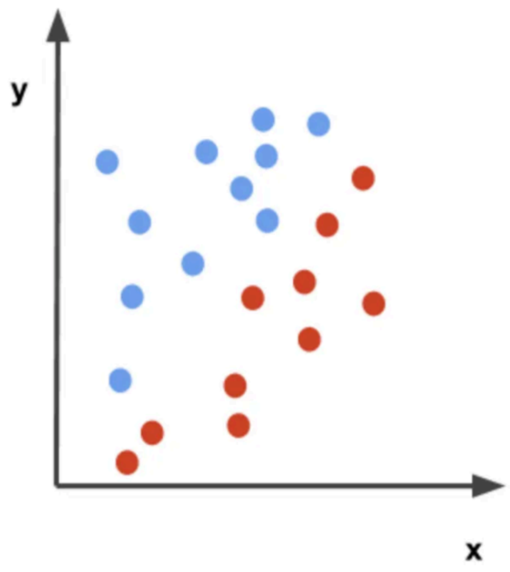
WHY PERCEPTRON IS USED

Perceptrons are the building blocks of the neural networks.It is typically used for the binary classifiers. For example let's take a simple perceptron.In this perceptron we have an input x and  y which are multiplied with weights wx and wy respectively, it also contains a bias. Lets create a graph with two different categories of the data represented with red and blue dots.Suppose our goal was to separate this data so that there is distinction between the blue and red dots. So how is this task done by a perceptron? A perceptron can create a decision boundary for a binary classification, where a decision boundary is regions of space on a graph that separates different data points

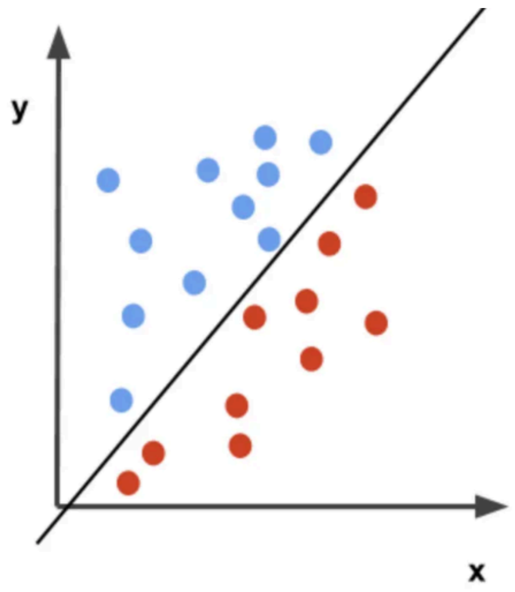
Lets play with the function to better understand this wx = -0.5

wy = 0.5 and b = 0  Then the function for the perceptron looks like this,

$$0.5x + 0.5y = 0$$



Before

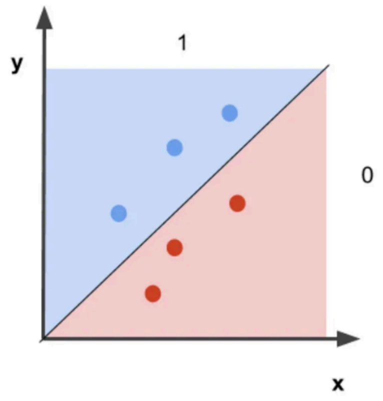


After

Let's suppose there is an activation function that outputs either 0 or 1. The perceptron function will then label the blue dots as 1 and the red dots as 0 in other words

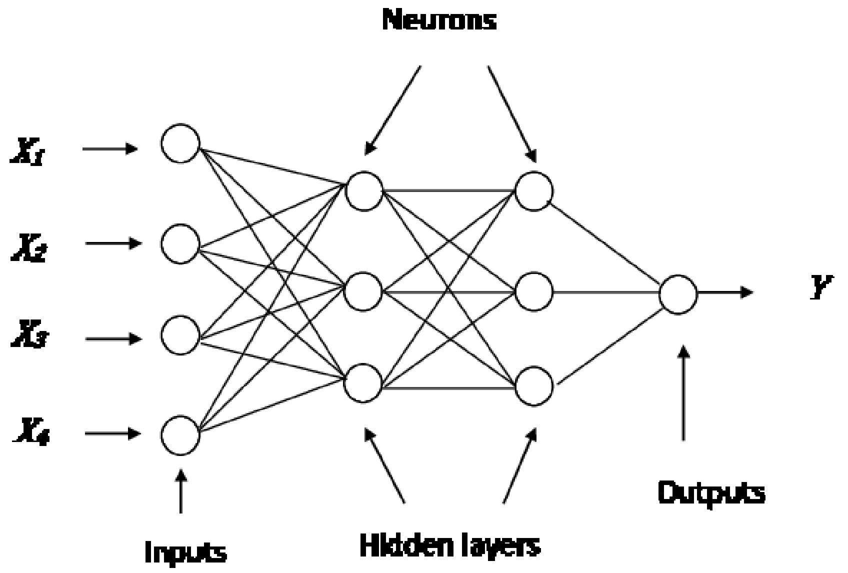
if 0.5x + 0.5y => 0, then 1

if 0.5x + 0.5y < 0, then 0



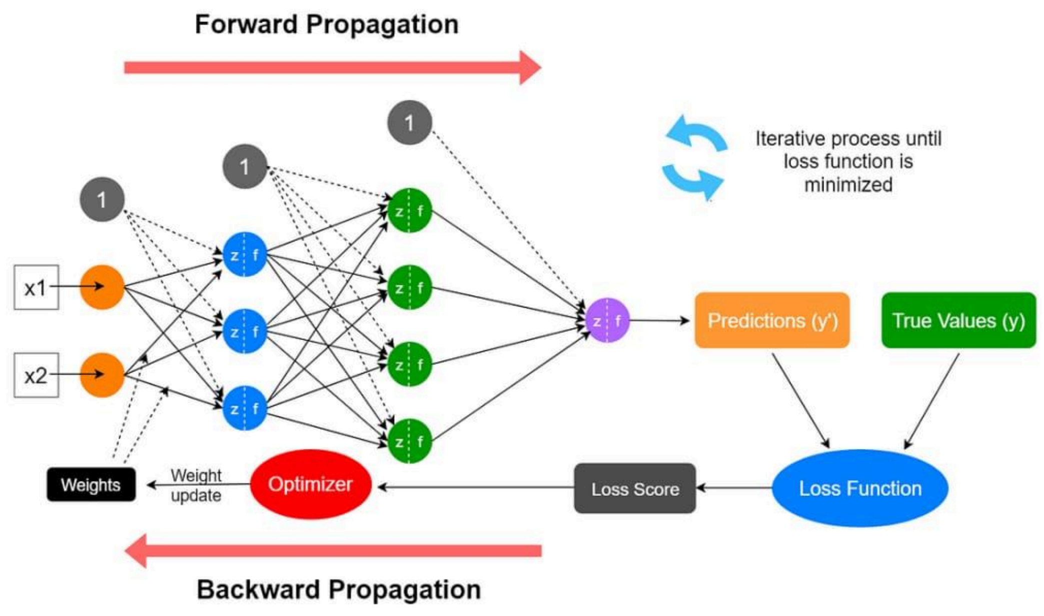
## 2. ARTIFICIAL NEURAL NETWORKS(ANN)

*Multilayer Perceptron* is also known as Artificial Neural Networks consists of more than one perceptron which is grouped together to form a multiple layer neural network(multiple hidden layers)



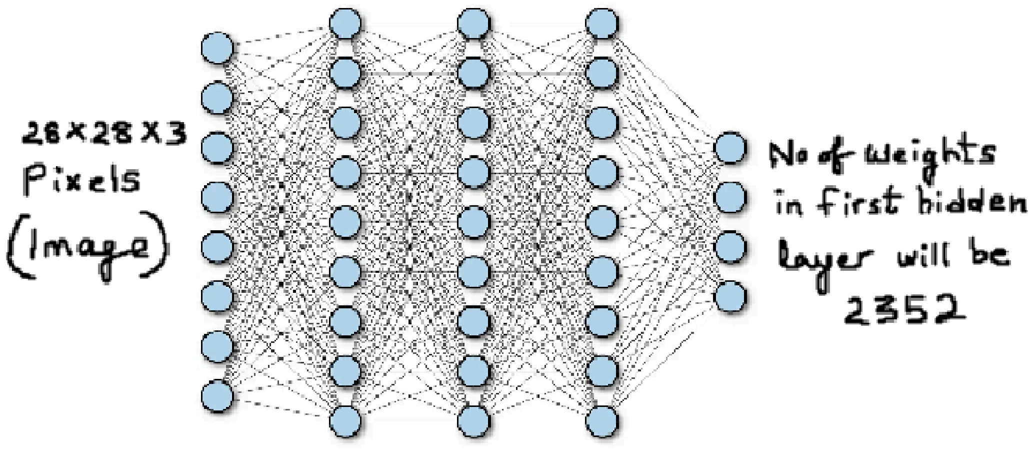
In the first step, Inputs are passed to the neural network with some weights attached to it to the hidden layer. We can have any number of hidden layers. Each hidden layer consists of neurons. All the neurons are connected to each neuron that's why it is also called a *Fully connected layer.* After passing on the inputs, all the computation is performed in the hidden layer. Then after passing through every hidden layer, we move to the last layer i.e output layer which gives us the final output this process can be explained as *Forward Propagation* We basically give the inputs which are called the input values and we also have the True values which should be the output of the input value. After computing in the dense layer it gives a value called predicted value it checks the difference between the true value and predicted value which is called Error or Loss if it is calculated for the one input whereas if the error is calculated for the bunch of inputs it is called Cost of the network.

If the Error is large then the steps taken to minimize the error and for the same purpose *Backpropagation* is performed and after the calculation of gradients in the backpropagation the network weights are updated using the optimizers for example *Gradient descent*

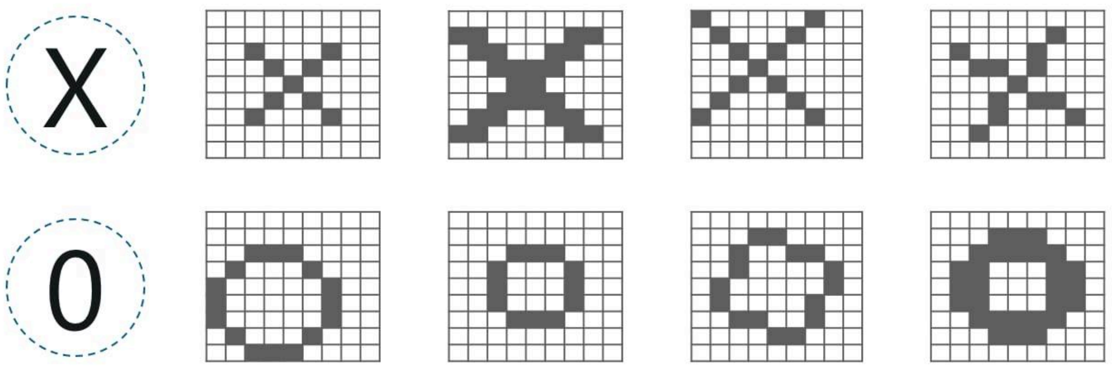


WHY NOT USE ANN ?

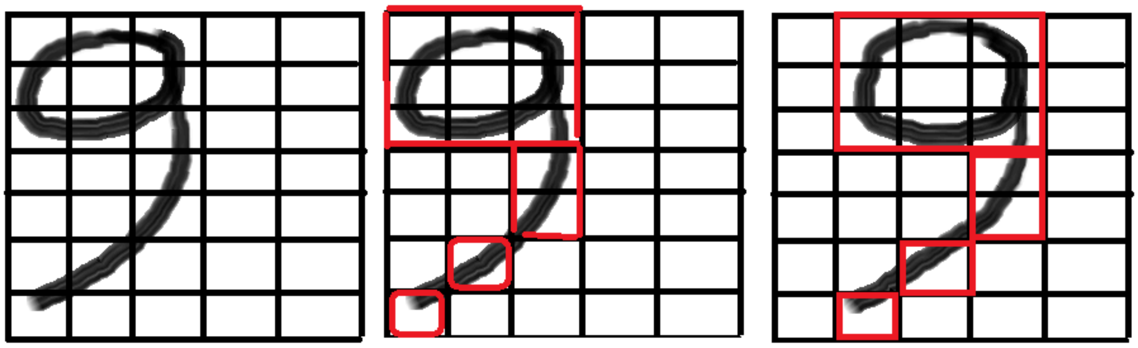
1. ANN's can properly process the textual data or csv file data but it is not good with image data. Image is represented as a group of pixels arranged in rows and columns and can be represented as a matrix for the computer to understand it. If we want to pass an image to the ANN network, we first *flatten* it, means converting these row and column matrix into 1D array i.e *column vector* for example if we take an image of size (28, 28) and we pass this to the network then we flatten it to the shape of (784 , 1) as ANN's operate on 1D vector where each input feature corresponds to a single neuron in the input layer. Each pixel value is assigned to one of the 784 input neurons in the ANN's input layer but usually when we take a picture in our mobile it has large pixel values when we send this image to ANN then it becomes very difficult for the network as the input neurons are more and so we will have more neurons in the hidden layers also so calculating the weights of so many neurons becomes tedious task as computational cost and processing time will be more. We have only talked about 1 image but in practice we send thousands of images from the dataset to the network which sounds crazy for the ANN to process it., this is one major drawback of ANN's



2. The second problem is there are different representations or versions of images for example if we take 2 alphabets X and O there are different ways one can write them like the images below



So although the 4 diagrams represent the same letter there are different representations of the image i.e  image may lie at any corner of the 28 x 28 pixel grid this is termed as *Spatial awareness* .Putting it together the information of the image is very important if there is an image of "X" model should know which pixels actually contain the information related to that X, But in ANN before processing the image we have to convert our 2D vector into 1D vector where all the pixels which have original/actual  information will get dispersed and the information may be lost



LOCATION SHIFTED

Pixel location shifted of an image may yield less accuracy when processed with ANN.Finally ANN is not preferred with handling the images as

1 . Too much computation cost

2. Treats local pixels same as the pixels far apart

3. Sensitive to location of an object in an image

**Image size = 1920 * 1080 * 3**

**First layer Neurons = 1920 * 1080 * 3 ~ 6 million**

**Hidden layer neurons = let's say you keep it ~ 4 million**

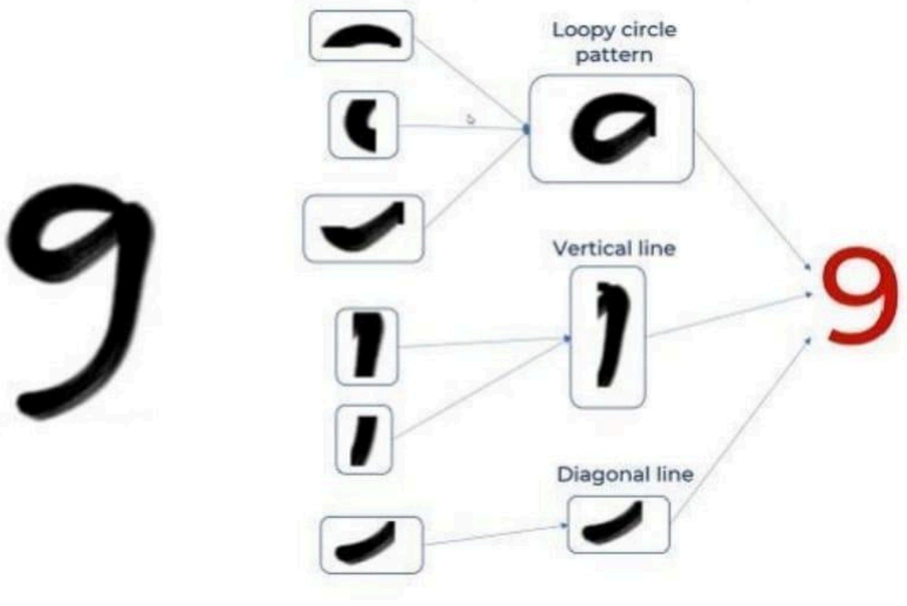
**Weights between input and hidden layer = 6 mil * 4 mil = 24 million**

CONVOLUTION NEURAL NETWORK(CNN)

A Convolutional neural network is a network architecture for deep learning *which learns directly from the data.* CNN's are particularly useful for finding the patterns in images to recognize objects. They can also be quite effective for classifying non-image data such as audio, time-series and signal data

| Aspect | ANN | CNN |
|---|---|---|
| Architecture | Fully Connected layers where each neuron connects to all the neurons on the next layer | Incorporates convolutional layers, pooling layers and fully connected layers |
| Input Data | Works well with structured data(eg:- tabular data) or low-dimensional data | Designed for spatial data,particularly images, videos or data with grid like topology |
| Feature Extraction | Relies on  manual feature extraction | Automatically extracts hierarchical features using convolution layers |
| Parameter sharing | Does not share weights across neurons, leading to a high number of parameters. | Shares weights in convolutional layers, reducing the number of parameters and improving efficiency. |
| Use Cases | ● Predictive modeling<br>● Financial forecasting<br>● Tabular data analysis | ● Image classification<br>● Object detection<br>● Facial recognition<br>● Natural language processing |
| Performance | Performs well on small to medium- sized datasets with simple relationships. | Excels on complex datasets, especially large- scale ones with spatial or hierarchical patterns. |
| Complexity | Relatively simple structure: easier to train on small datasets. | More complex requires more computational resources and often larger datasets for effective training. |

In CNN if an image needs to be identified it divides the image into different parts or sections and processes it , If we take an image of “ 9 “ first layer identifies the parts of the independent features and the second layer identifies the loopy patterns, vertical and diagonal line patterns and concludes the given image is nine by combining all the features. So what are these layers which identify the features in the image? These layers are called *Convolution layers*.

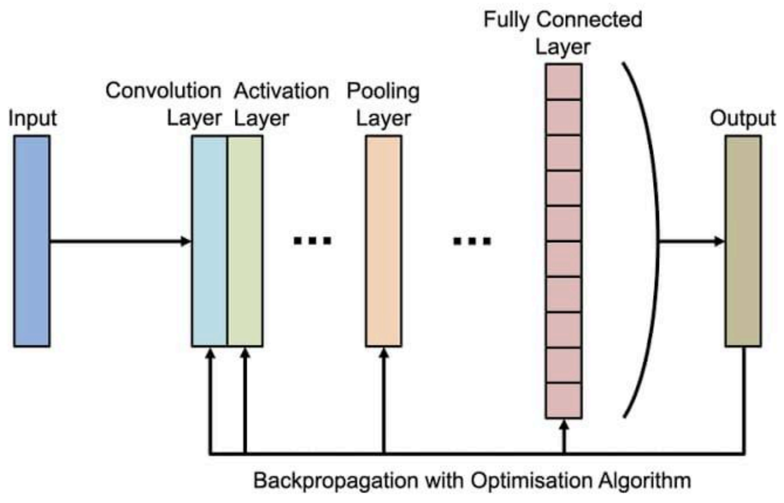


**ARCHITECTURE OF  CNN**

The standard architecture of CNN is composed of a series of layers, each serving a specific purpose

Basic layers in a CNN architecture

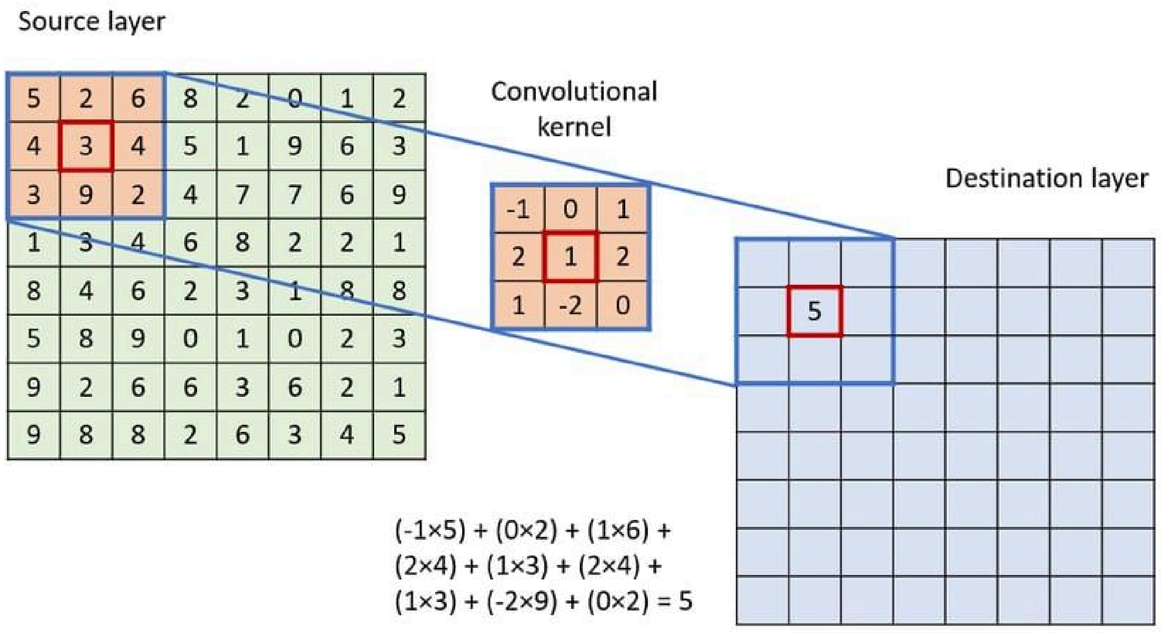
1. Input layer
2. Convolution layer
3. Activation layer (eg : ReLU)
4. Pooling layer (eg :Max pooling)
5. Fully Connected layer
6. Output layer



## 1. CONVOLUTION LAYER

It is the core and the most important layer while processing the image and extracting the key features of an image

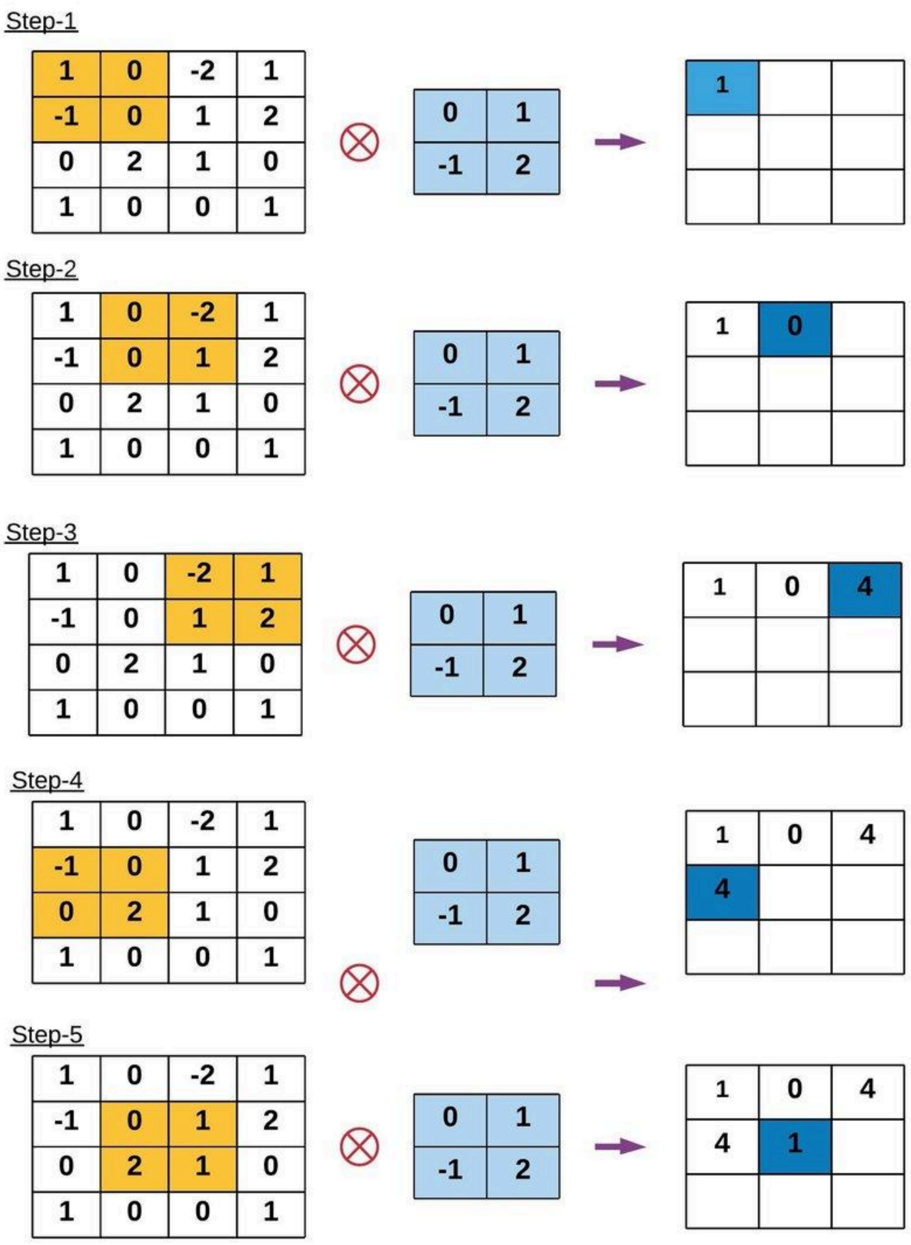
1. The core layer in CNNs, this layer applies convolutional filters (or kernels) over the input image to detect patterns.(kernels are randomly generated weights that slide over the image to extract features)
2. Each filter slides (convolves) over the image and computes a "dot product" between the filter and a small section of the input, generating a *feature map.*
3. Each filter is designed to detect specific patterns like edges, textures, or shapes.



To apply the convolution over an image:

1) Overlay the *Kernel* on the Image: Start from the top-left corner of the image and place the kernel so that its center aligns with the current image pixel.
2) Element-wise Multiplication: Multiply each element of the kernel with the corresponding element of the image it covers.
3) Summation: Sum up all the products obtained from the element-wise multiplication. This sum forms a single pixel in the output feature map.

4) Continue the Process: Slide the kernel over to the next pixel and repeat the process across the entire image.



Hyperparameters:

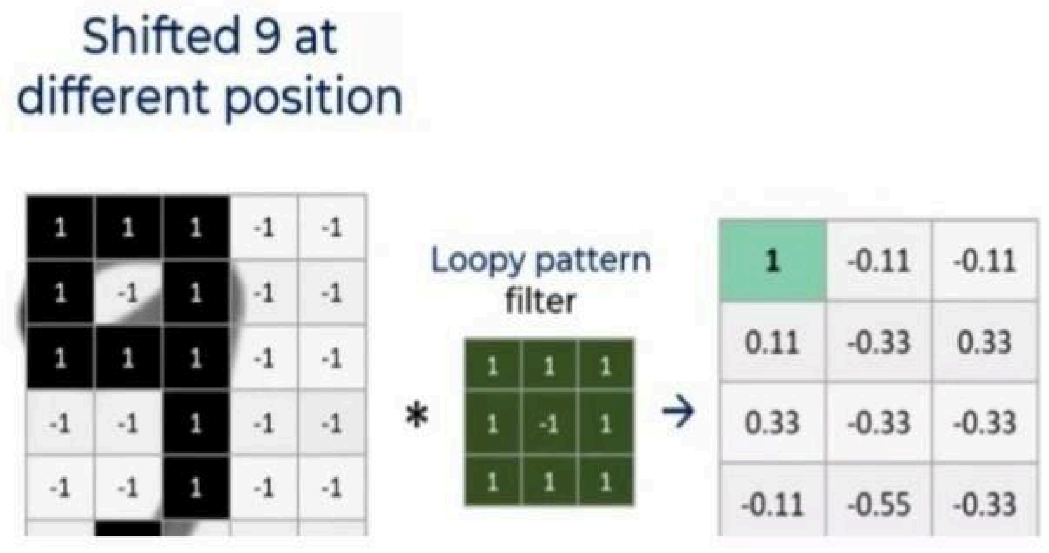
1.*Filter Size (Kernel Size)*: Usually 3x3 or 5x5.

2.*Stride*: Determines the step size as the filter slides over the input.

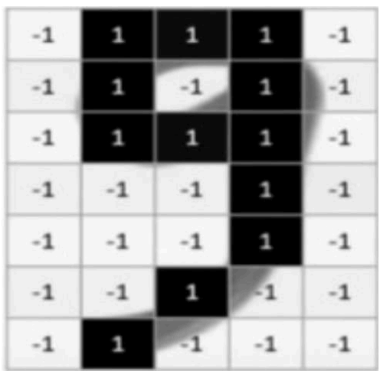
3.*Padding*: Controls the spatial dimensions of the output (adding "zero-padding" can preserve input size).

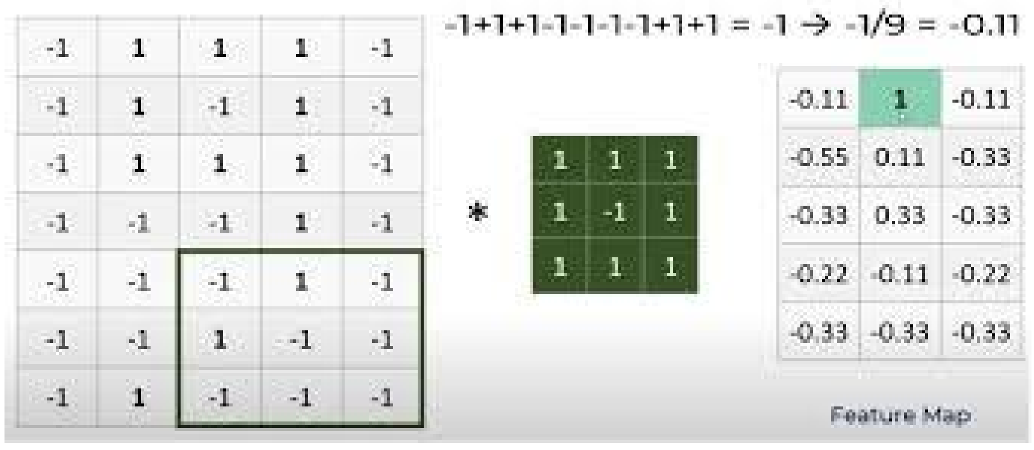
Output:

The output of a convolutional layer is a set of feature maps, each representing the presence of a specific feature across the spatial dimensions.

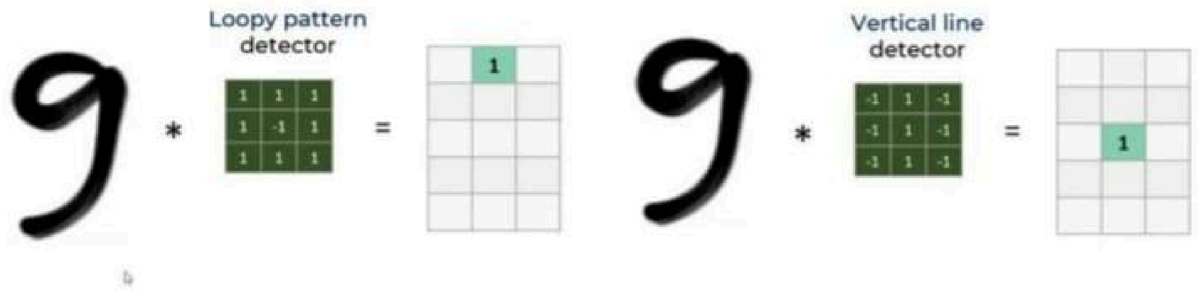


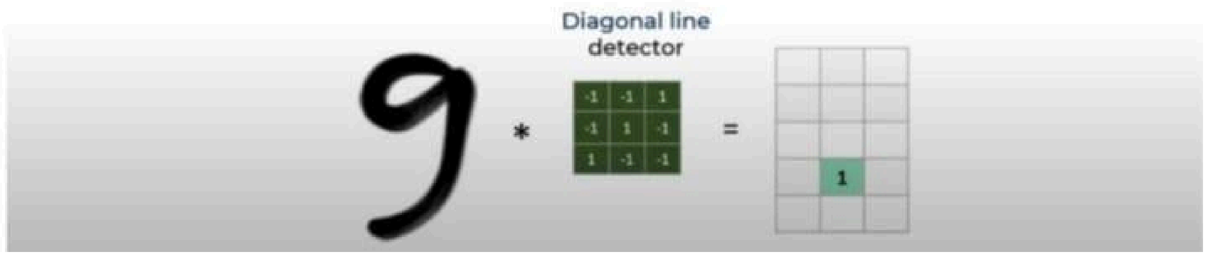
The loopy pattern of the image “9” is present in the top left of the image so the by applying filter we got 1(most prominent) on the feature map which means the location where 1 is present indicates the location of loopy pattern in the image, Even when “9” is shifted to middle it gets identified in the feature map



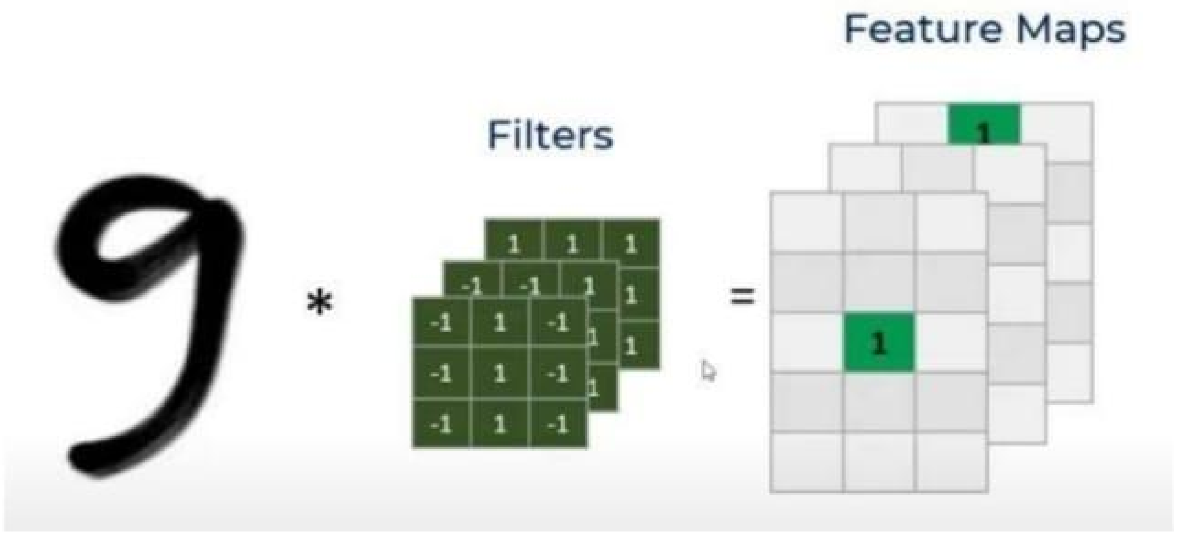


This is how it detects the digit



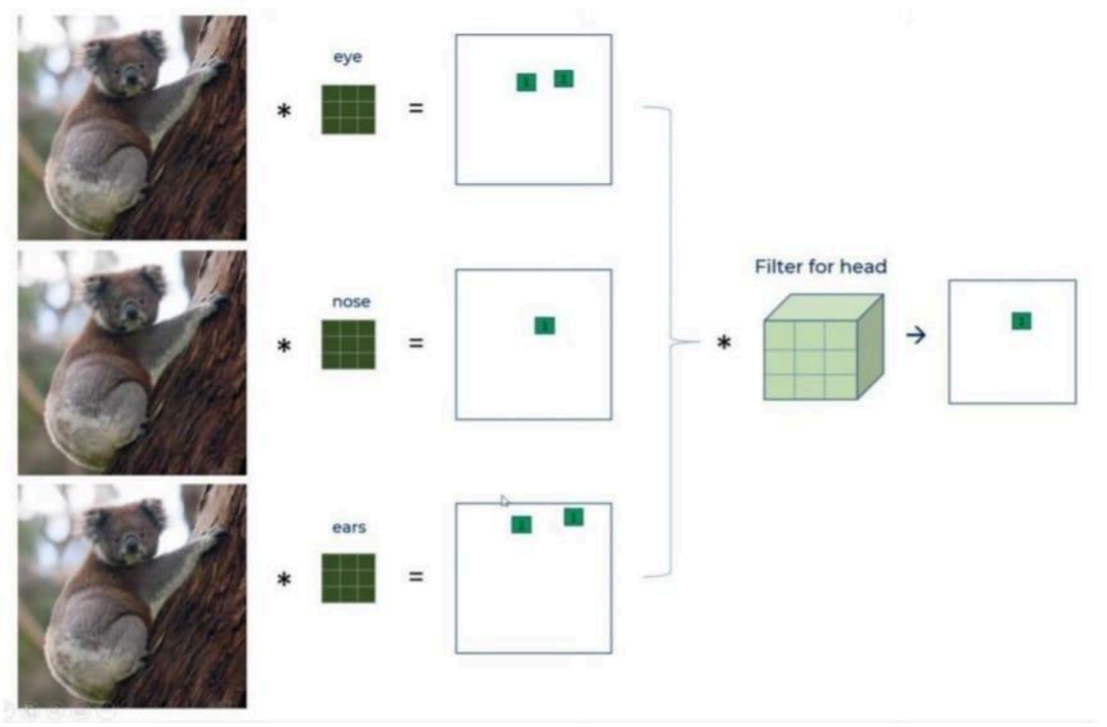


By combining all the feature maps of that image it gives the output

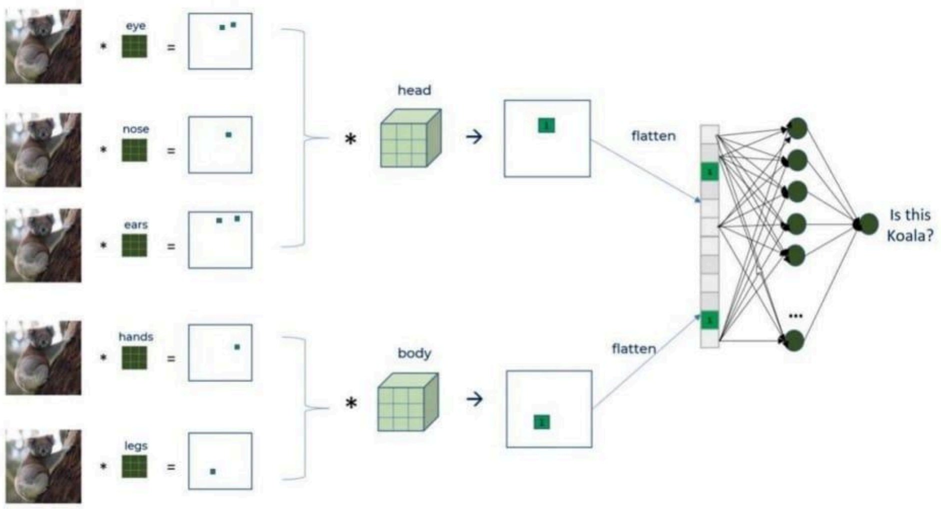


The same goes for an image which contains any background, animal, objects etc.Take an example of koala bear which is identified by its nose, eye, ears, hands, legs by different convolution layers

We first take the example of koala bear head where eye, nose and ears are identified through this layer



As per the above figure , we are aggregating the results using the different filters for the head, and it gives the featured map of the Koala head detector. Similarly, the Koala body detector for body detection featured a map. Finally, we will flatten the one's which are available in the featured maps of the head and body of the Koala, which means converting 2D array to 1D array and join them together to get a fully connected dense neural network for classification. In the case of the same Koala in a different form, the neural networks are used to handle the variety in your inputs. Such that, it can generically classify that variety of inputs. In CNN, feature extraction and classification take place.



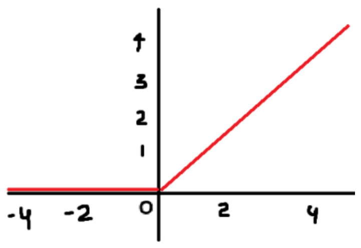
```
class Convolution:

    def __init__(self, input_shape, filter_size, num_filters):
        input_height, input_width = input_shape
        self.num_filters = num_filters
        self.input_shape = input_shape

        # Size of outputs and filters

        self.filter_shape = (num_filters, filter_size, filter_size) # (3,3)
        self.output_shape = (num_filters, input_height - filter_size + 1, input_width -
filter_size + 1)

        self.filters = np.random.randn(*self.filter_shape)
        self.biases = np.random.randn(*self.output_shape)
```

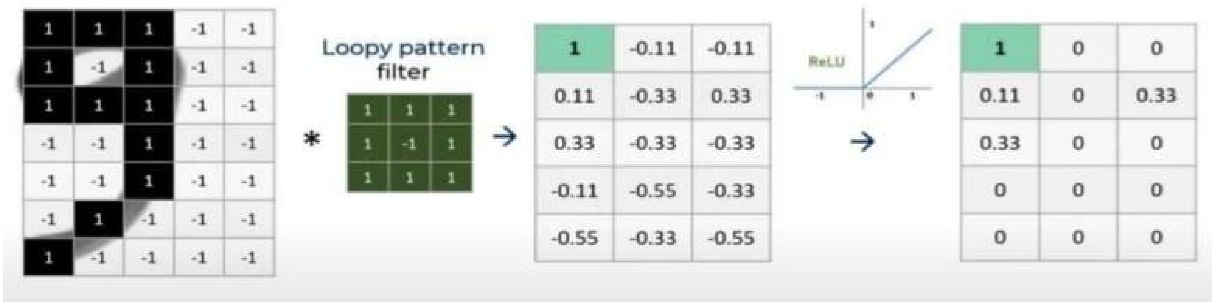
Code snippet for the Convolution layer Forward pass

2. ACTIVATION LAYER

1. Following the convolutional layer, an activation function is typically applied element-wise to add non-linearity to the network.
2. The Rectified Linear Unit (ReLU) is the most common activation function in CNNs. It replaces all negative values with zero, keeping only the positive activations.
3. This layer helps the CNN learn more complex patterns by allowing non-linear combinations of features.

The corrected linear activation function, or ReLU, is a piecewise linear function that outputs the input directly if the input is positive and 0 otherwise. Because a model that utilizes it is quicker to train and generally produces higher performance, it has become the default activation function for many types of neural networks, all the negative values are replaced with zero (0). The values are more than zero, they will keep as it is. ReLU helps with making the model non-linear  f(x) = max(0, x)



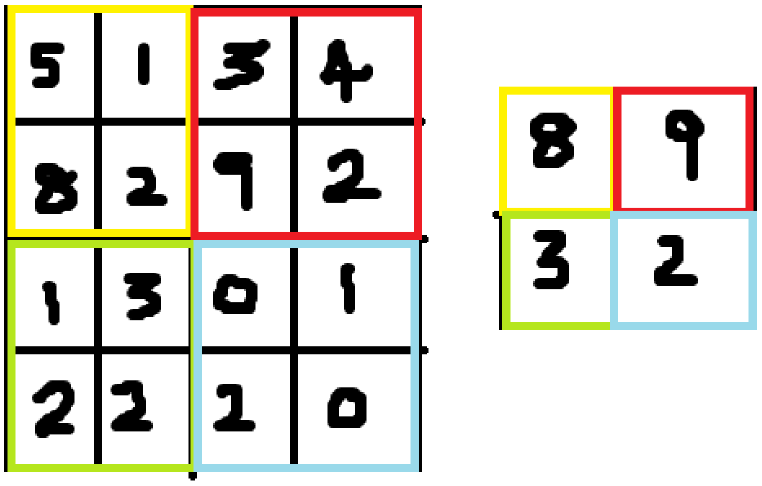


After getting a feature map other extra features that we don't want get eliminated by using ReLU activation function

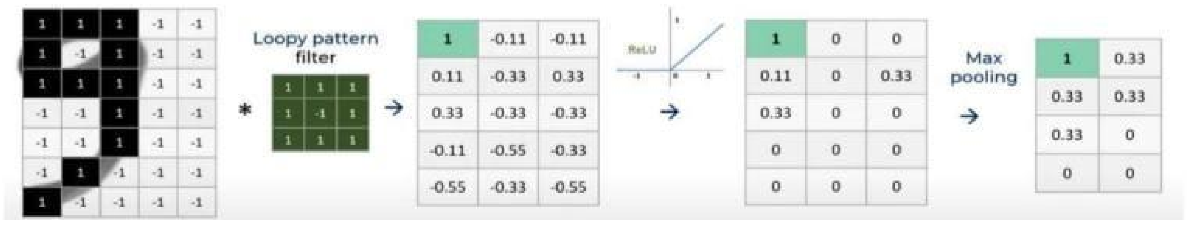
3. POOLING LAYER(MAX POOLING)

"Pooling layer is basically used to reduce the size of an image"

1. Pooling layers reduce the spatial dimensions (height and width) of the feature maps, which decreases the number of parameters and computational load.
2. Max Pooling is the most common pooling method, where the maximum value within a sliding window (e.g., 2x2) is taken to represent that region.
3. Pooling also adds a degree of translation invariance, making it easier for the network to recognize objects in different positions within the image.

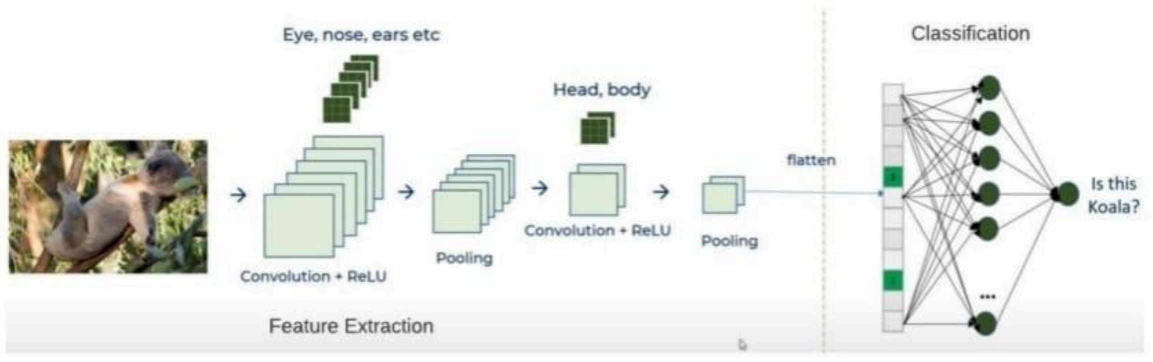


2 x 2 filter with stride 2



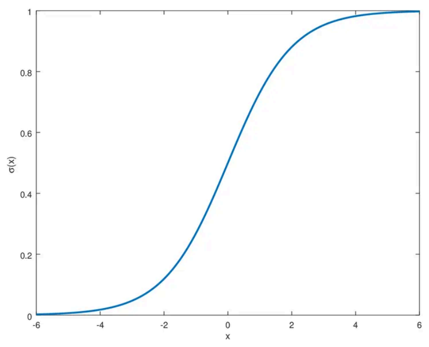
```
import numpy as np

class MaxPooling:
    def __init__(self, pool_size):
        self.pool_size = pool_size

    def forward(self, input_data):
        self.input_data = input_data
        self.num_channels, self.input_height, self.input_width = input_data.shape

        # Calculating output dimensions
        self.output_height = self.input_height // self.pool_size
        self.output_width = self.input_width // self.pool_size

        # Initializing the output with zeros
        self.output = np.zeros((self.num_channels, self.output_height, self.output_width))

        # Performing max pooling
        for c in range(self.num_channels):
            for i in range(self.output_height):
                for j in range(self.output_width):

                    start_i = i * self.pool_size
                    start_j = j * self.pool_size
                    end_i = start_i + self.pool_size
                    end_j = start_j + self.pool_size

                    # Extract patch and apply max pooling
                    patch = input_data[c, start_i:end_i, start_j:end_j]
                    self.output[c, i, j] = np.max(patch)

        return self.output
```

Finally after all these operations the pooled output is sent to the fully connected layer after flattening the array to get the final output of the image as koala or not(in this example)



4. OUTPUT LAYER:

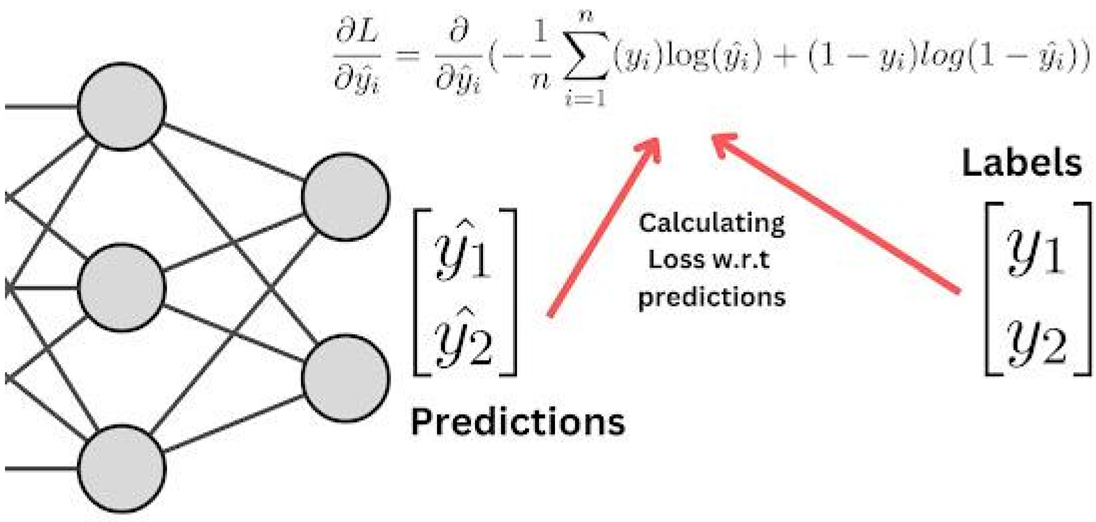
The output layer depends on the task at hand:
1. Multiclass Classification: A softmax activation function is often used in the output layer for multi-class classification. The softmax outputs a probability distribution over classes, with each value representing the probability of the input belonging to a specific class.
2. Binary classification: Sigmoid activation function is usually used in the output layer for binary-classification because for any given value it gives values ranging between 0 and 1 which makes it useful. Outputs can be interpreted as probabilities thus making it suitable for binary classification
When the input is large(positive) output approaches 1, if the input is small(negative) output approaches 0 and when the input is 0 it gives exactly 0, More about activation functions
Activation Functions.pdf



The loss Function used in the Fully connected layer is Binary Cross entropy(BCE)

Binary Cross-Entropy (BCE) is used in binary classification because it quantifies the difference between predicted probabilities and true binary labels (0 or 1). It ensures the model learns to output probabilities close to the actual labels by penalizing incorrect predictions more heavily. BCE is ideal for optimizing models in binary classification tasks as it directly minimizes the log loss. Here is the mathematical derivation of the BCE-derivative and its compatibility with sigmoid activation function BCE-LOSS derivative.pdf

$$BCE = -\frac{1}{n}\sum_{i=1}^{n}(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))$$



$\frac{\partial L}{\partial \hat{y}_i} = \frac{\partial}{\partial \hat{y}_i}(-\frac{1}{n}\sum_{i=1}^{n}(y_i)\log(\hat{y}_i) + (1 - y_i)log(1 - \hat{y}_i))$
$\begin{bmatrix}\hat{y}_1 \\ \hat{y}_2\end{bmatrix}$
Predictions
Calculating Loss w.r.t predictions
Labels
$\begin{bmatrix}y_1 \\ y_2\end{bmatrix}$

After getting the loss we go backward from the fully connected layer to the input layer by *backpropagation* where we calculate the gradient of loss function with respect to the parameters and adjust the parameters using the updation rule using any optimization technique we want.In fully connected layer we change the weights and biases of the layer by their gradients and then calculate the gradients in the convolution and pooling layer, Here is the brief overview of the process if you want to see the maths behind it take a look at Backpropagation_maths.pdf

Fully Connected Layer:

- Compute the gradient of the loss with respect to the outputs of the fully connected layer.
- Calculate the gradients with respect to the weights and biases of this layer using the chain rule.

- Propagate the error gradients back to the preceding layer (usually a flattened layer).

Flattening Layer:

- Reshape the gradients from the fully connected layer into the shape of the output feature maps of the last convolutional layer.
- Pass these gradients backward to the last convolutional layer.

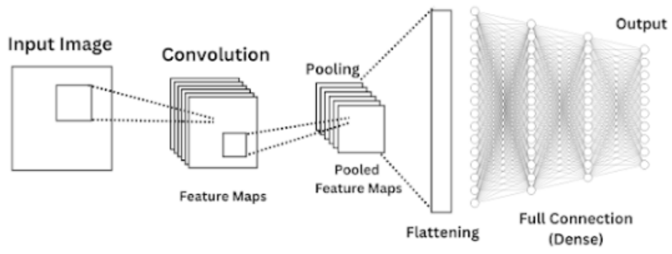
Convolutional Layers:

- For each convolutional layer, calculate the gradient of the loss with respect to its output feature maps.
- Use these gradients to compute the gradients with respect to the filters (kernels) by convolving the input to the layer with the error gradients.
- Propagate the error further back to the previous layer by convolving the error gradient with the kernels.

and the optimizer used in our project is *SGD(Stochastic gradient descent) with momentum* You can visit this document OPTIMIZERS(first_Edition) (1).pdf for more understanding of stochastic gradient descent and also about other optimizers

## V INTEGRITY OF THE SYSTEM

We developed this CNN Architecture using Pytorch and there are also quite few other libraries we used in this project such as, matplotlib for plotting the results or even checking if the images with labels are correctly loaded, scikit learn for splitting the dataset into train and test and also for performance graphs , PIL for loading the images, and some preprocessing work.The dataset we used contains 5868 train images and 1680 test images and the architecture of the layers is as follows, 2 convolution layers with 32, 64 filters(kernel_size = 3) with *ReLU activation function* and 2 pooling layers (1 pooling layer after each convolution layer) then coming to fully connected layer we used 2 dense layers with *dropout* of 50% dropout rate for *regularization* and at last a *sigmoid activation function* for binary output is used along with *BCE loss function*. If you want to know more about dropout layer you can visit this document which contains the brief understanding of dropout regularization



Working in the fields of deep learning, computer vision, and convolutional neural networks often requires powerful hardware and specialized software to process large amounts of data and perform complex computations. As technology continues to evolve, it is important to keep up with the latest hardware and software advancements to ensure that our systems are efficient and effective. To that end, we have designed our pothole detection system with upgradability in mind. Our system is designed to be modular, which allows for easy hardware and software upgrades as new technology becomes available. By staying up to date with the latest tools and technologies, we can ensure that our system remains accurate, reliable, and efficient. Overall, we believe that it is essential to stay ahead of the curve when working in these fields, and we are committed to continuously upgrading and improving our system to keep up with the latest advancements in hardware and software.

# VI RESULT DISCUSSION

## A. Evaluation and Detection Rate

We included quantitative and qualitative measures to assess the effectiveness of our pothole-detection technology. Accuracy, precision, recall, and F1 score were among the quantitative measurements; user feedback and visual inspection were among the qualitative ones. We employed a test set of road photos that had been manually labelled as either containing potholes or not to determine the accuracy, precision, recall, and F1 score. On this test set, we next ran our pothole detecting algorithm, and we compared the predicted labels to the actual labels. The accuracy represents the frequency with which the predicted label matches the actual label, the precision the percentage of actual potholes among the predicted ones, and the recall the percentage of actual potholes that the system properly identified. The F1 score is the harmonic mean of precision and recall, and provides a balanced measure of both

In order to confirm the precision and thoroughness of the detected potholes, we also performed visual inspections of them. Finally, we obtained user feedback on the system's applicability and usability in real-world scenarios from individuals like road maintenance employees.

The quality of the input photos, the complexity of the road environment, and the threshold for identifying whether a detected object is a pothole or not all affect the detection rate of our proposed pothole identification system.

In our tests, our model has reached a detection rate of 85.7%. This indicates that, while reducing false positives, our algorithm successfully identified 85.7% of the potholes in the test set. Additionally, we attained a high F1 score of 0.865, which denotes a harmony between recall and precision. It is crucial to remember that the detection rate can change based on the particular driving conditions and the caliber of the input photos. For instance, the detection rate may drop if the photos are pixelated or fuzzy. Additionally, if the road environment is congested or highly complex, the algorithm can have a hard time telling potholes apart from other things like cracks or shadows. Overall, our findings show the potential of CNN-based pothole detection systems to enhance traffic safety and infrastructure upkeep, but additional study is required to assess how well they work under various real-world scenarios.

(i) Precision: It is the proportion of correctly classified potholes (true positives) to the total number of instances predicted as potholes (true positives + false positives).

The formula for precision is given as:

$$precision = \frac{correctly\ classified}{total\ predicted}$$

In our case Precision = 0.809(80.9%)

(ii) Recall:- It is used to measure how well the CNN algorithm correctly identifies the proportion of actual potholes (true positives) out of all the potholes that exist in the dataset (true positives + false negatives). The formula for recall is given as

$$recall = \frac{correctly\ classified}{total\ actual}$$

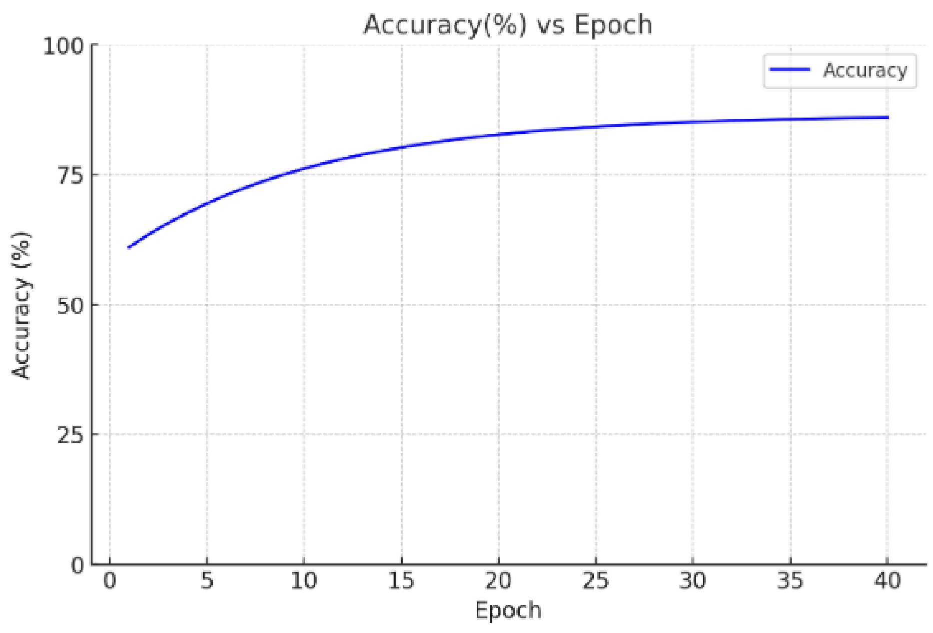
In our case Recall = 0.933(93.3%)

(iii) F1-Score: This is a single score that represents precision and recall. The formula is given as:

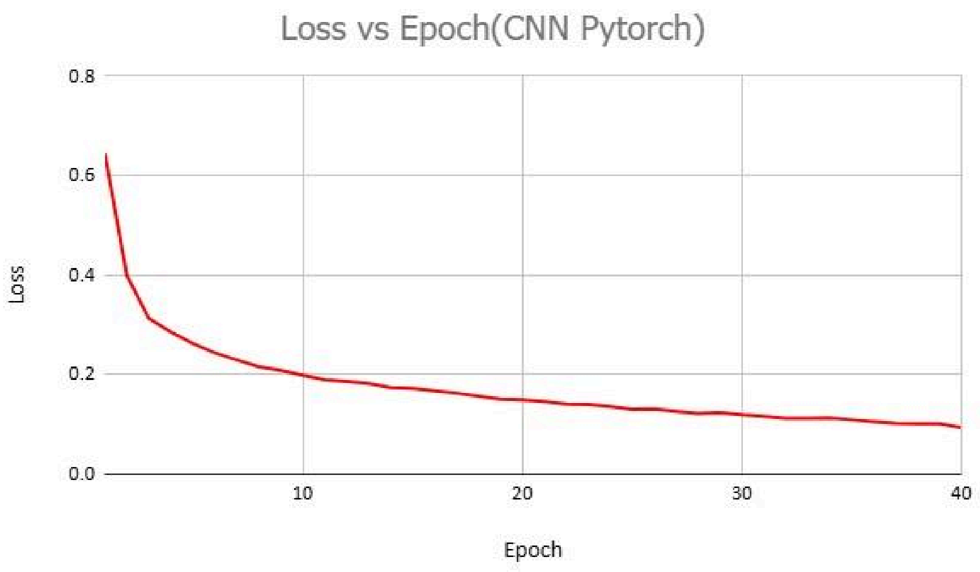
$$F1\ score = \frac{2 \times precision \times recall}{precision + recall}$$

In our case F1-score=0.865(86.5%)

Training Benchmark graphs



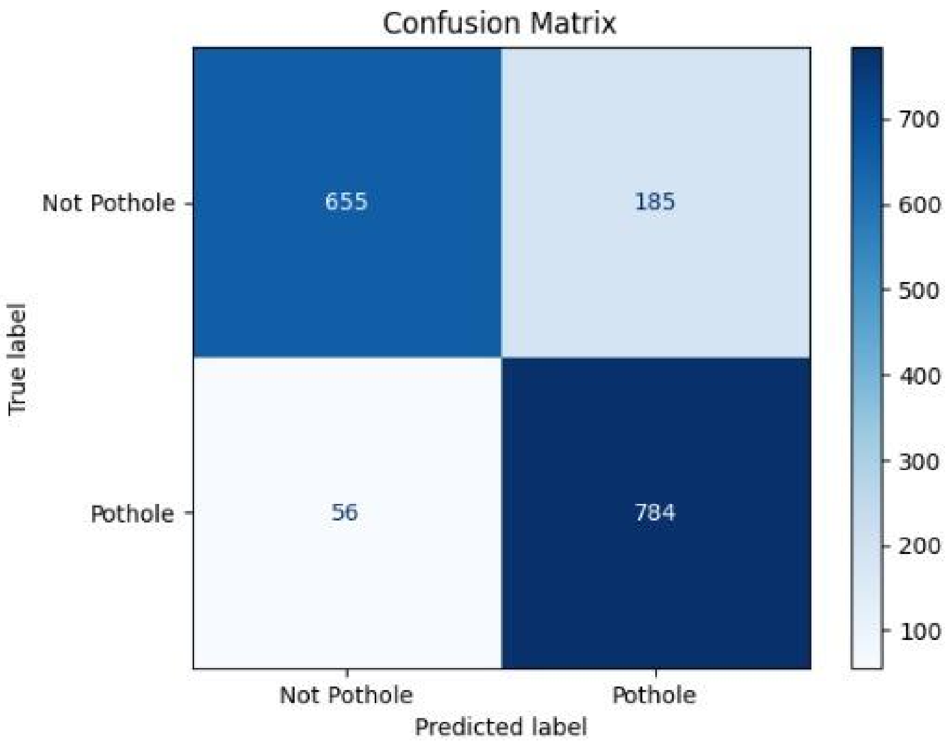
Model Accuracy curve



Model Loss Curve

(iv) Confusion matrix:-

In the confusion matrix, the rows show the actual value and the columns, the predicted values.



Confusion matrix

True Positives (TP): 784 potholes were correctly identified as potholes.

True Negatives (TN): 655 non-pothole cases were correctly identified as not potholes.

False Positives (FP): 185 non-pothole cases were misclassified as potholes.

False Negatives (FN): 56 potholes were misclassified as not potholes.

## B. Advantages Of Work

One of the main advantages of our approach is that it allows for the observation of potholes in a better and more effective way by providing live video capturing. This enables real-time monitoring and detection of potholes, which is crucial for ensuring road safety. Moreover, with the help of CNN, detecting potholes has been done in an appropriate manner, which mirrors growth and development towards the field of Self Driving Cars in harsh conditions.

## C. How to Improve the Accuracy

There are a number of potential areas for improvement that can be investigated in order to enhance the pothole detection system's efficacy further:

**Data augmentation**: Increasing the quantity and diversity of the training dataset through the use of data augmentation techniques like flipping, rotating, or zooming may enhance the accuracy and resilience of the CNN model.
**Transfer learning**: This technique allows a pre-trained model to be improved upon using a particular dataset. Using a pre-trained model allows for speedier fine-tuning than starting from scratch and allows the model to learn more effectively with less input. To enhance the pothole detection system's performance, this technique can be investigated.
**Hyperparameter tuning**: The selection of hyperparameters, such as learning rate, batch size, and regularization, has a significant impact on the CNN model's performance. The  model's performance can be further improved by employing methods like grid search or random search to investigate various combinations of hyperparameters.

# VII SPOTHOLE PLATFORM

Potholes are a major inconvenience while traveling on roads, especially in India. They cause damage to vehicles and even accidents. Identifying and addressing potholes is important for maintaining road safety and overall infrastructure. However, traditional methods of pothole detection, which often rely on manual inspection, are time-consuming, labor-intensive, and tiring. This calls for the need for an efficient way to deal with this problem. Although our CNN-based pothole detection model is successful in predicting potholes, to make it more accessible and easy to use for a layman, we have designed a web interface to handle this issue. The web interface allows users to take advantage of the model's capabilities and also provides some additional functionality, which offers convenience to the users and aids in an overall better experience.

The SPotHole platform is an innovative web application built using the MERN (MongoDB, Express, React, Node.js) stack by combining the power of machine learning, and Web Development tools, SpotHole aims to simplify the process of identifying and managing road defects

**Main Features of the SPotHole Platform**

- **Login/Signup** The SPotHole platform ensures a secure and personalized user experience by providing login and signup functionality. New users can create an account by signing up with their email address and a secure password. Existing users can log in to access their dashboard, report potholes, and track their submissions. This authentication system, powered by JWT, ensures data privacy and session management.
- **Reporting Potholes** Users can easily report potholes by uploading images through the web interface. The uploaded images are processed by the CNN model, which detects potholes ,the platform also collects relevant metadata, such as the location of the pothole, this can potentially provide actionable insights for road maintenance teams.
- **Maintaining a History of Reported Potholes** SootHole allows users to maintain a detailed history of all the potholes they have reported. This feature enables users to track the status of their submissions, the history is presented in an intuitive and organized format, ensuring ease of use.

- **Changing User Credentials** The platform provides users with the flexibility to update their account credentials, such as email address and password. This feature ensures that users can keep their account information up to date and maintain security. The process is straightforward and secure, enhancing the overall user experience.

**Using the MERN Stack to Design SPotHole**

The SPotHole platform is designed using the MERN stack, which provides an efficient framework for web development. Here's how each component of the MERN stack is utilized in the platform:

- **MongoDB**: MongoDB is used as the database for storing all the application data, such as user details and pothole reports,Its NoSQL nature allows for flexibility in data storage and retrieval, making it ideal for handling dynamic data like user submissions and image metadata.
- **Express.js**: Express.js is the backend framework used to build the server-side logic of the application. It manages API routes for user authentication, pothole reporting, and data retrieval. The integration of Express.js with Node.js ensures seamless communication between the frontend and the database.
- **React.js**: React.js powers the frontend of SPotHole, offering a dynamic and interactive user interface. The component-based architecture of React ensures that the platform is modular and easy to maintain. Users can navigate through features like login/signup, pothole reporting, and history tracking with a smooth and responsive experience.
- **Node.js**: Node.js serves as the runtime environment for the server-side application. It handles incoming requests from the frontend, processes them, and interacts with MongoDB to fetch or update data.

The MERN stack's cohesive nature allows for seamless integration between the frontend, backend, and database, resulting in a user-friendly platform. By leveraging the strengths of each component, SpotHole is able to provide an efficient solution for pothole detection and reporting.

Below is the overall architecture of our project

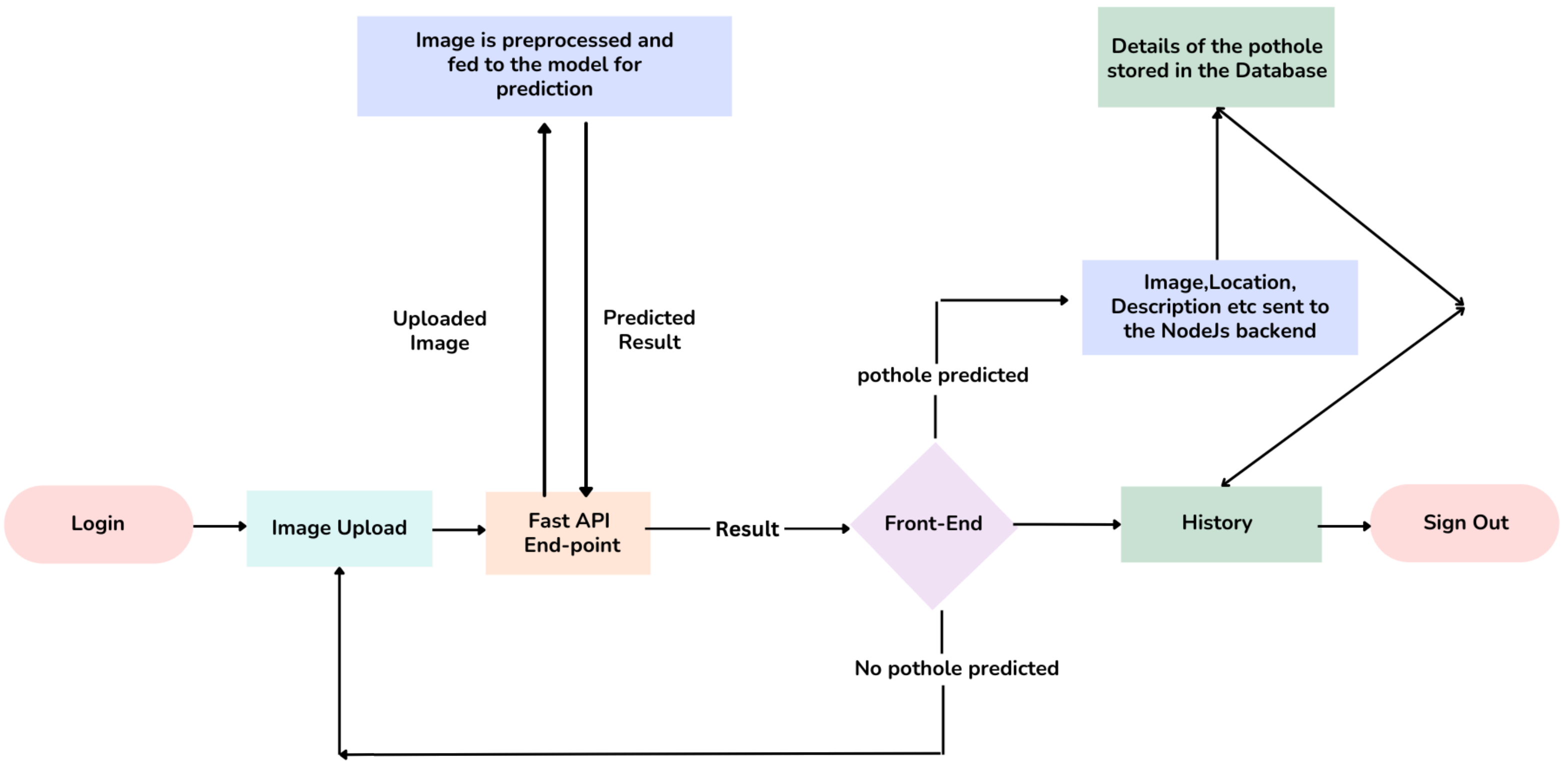


Image is preprocessed and fed to the model for prediction
Details of the pothole stored in the Database
Uploaded Image
Predicted Result
Image,Location, Description etc sent to the NodeJs backend
pothole predicted
Login
Image Upload
Fast API End-point
Result
Front-End
History
Sign Out
No pothole predicted

The backend of the application is built using Node.js with the Express.js framework. Several key modules are used to handle various functionalities such as request parsing, authentication, database interactions, and cross-origin resource sharing (CORS). Below is a detailed explanation of each module and its role in the backend

**1. Express :**

- express is a minimal and flexible Node.js web application framework that provides a robust set of features for building web and mobile applications.
- It simplifies the process of creating APIs, handling HTTP requests, and managing routes.
- Used to create the server and define routes for handling different endpoints (e.g., /api/v1/auth/signup, /api/v1/auth/login)
- Middleware functions (e.g., body-parser, cors) are integrated with Express to enhance its functionality.

**2. Body-parser :**

- body-parser is a middleware that parses incoming request bodies in a middleware before your handlers.
- It extracts the entire body portion of an incoming request stream and makes it available on req.body.
- Used to parse JSON and URL-encoded data sent in the request body (e.g., form data, JSON payloads).
- Essential for handling POST and PUT requests where data is sent in the request body.

**3. jwt (jsonwebtoken) :**

- jsonwebtoken is a library used to create and verify JSON Web Tokens (JWTs).
- JWTs are used for securely transmitting information between parties as a JSON object, often used for authentication and authorization.
- Used to generate tokens when a user logs in or signs up.
- Tokens are sent to the client and stored in sessionStorage.
- The token is included in subsequent requests to authenticate the user.

**4. bcryptjs:**

- bcryptjs is a library used for hashing passwords.
- It provides a secure way to store passwords by hashing them with a salt, making it difficult for attackers to reverse-engineer the original password.
- Used to hash passwords before storing them in the database.
- Used to compare hashed passwords during login to verify user credentials.

**5. mongoose Purpose:**

- mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js.
- It provides a schema-based solution to model application data, making it easier to interact with MongoDB.
- Used to define schemas and models for MongoDB collections (e.g., User,History). Provides methods for querying, updating, and deleting documents in the database.

**6. cors (Cross-Origin Resource Sharing) :**

- cors is a middleware that enables Cross-Origin Resource Sharing.
- It allows or restricts resources on a web page to be requested from another domain outside the domain from which the resource originated.
- Used to allow the frontend (running on a different origin, e.g., http://localhost:3000) to communicate with the backend (e.g., http://localhost:5000).
- Configures which origins, methods, and headers are allowed.

**Detailed breakdown of the architecture**

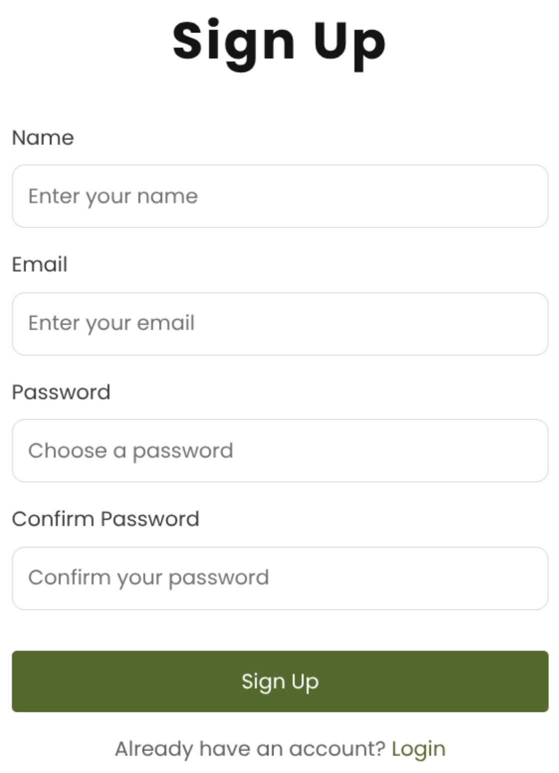
**Authentication :**

**1. Signup Workflow**

The signup process involves creating a new user account.

**Frontend Process**

- **User Interaction:**
    - The user enters their name, email, password into a signup form.
    - The user is supposed to re-enter the password for confirmation
- **Data Submission:**
    - Once the user fills out the form and clicks the "Sign Up" button, the frontend performs basic validation (e.g., checking if all fields are filled, if the passwords match, etc.).
    - If the validation passes, the frontend sends the user's details to the backend in the form of a JSON object via an API call. This is typically done using a POST request to the /api/v1/auth/signup endpoint.



**Backend Processing:**

**Receiving the Request:**

- The backend receives the request at the /api/v1/auth/signup endpoint. The request body contains the user's details (name, email, password, and confirmPassword), which are extracted using req.body.

**Input Validation:**

- The backend first checks if all required fields (name, email, password, confirmPassword) are present. If any field is missing, it responds with a 400 Bad Request status and an error message: "All fields are required".
- The backend then checks if the password and confirmPassword fields match. If they don't match, it responds with a 400 Bad Request status and an error message: "Passwords do not match".

**Checking for Existing User:**

- The backend queries the database to check if the provided email already exists in the Users collection.
- If a user with the same email already exists, the backend responds with a 400 Bad Request status and an error message: "Email already registered".

**Hashing the Password:**

- If the email is unique, the backend proceeds to hash the user's password using a library like *bcrypt*. Hashing ensures that the password is stored securely in the database. The password is hashed with a salt factor of 10 (e.g., await bcrypt.hash(password, 10)).

**Creating a New User:**

- A new user object is created using the Users model, with the name, email, and hashedPassword fields. The new user is saved to the database.

**Response to Frontend:**

- If the signup process is successful, the backend responds with a 201 Created status and a success message: "User registered successfully".

**Error Handling:**

- If any error occurs during the signup process (e.g., database connection issues, validation errors), The backend logs the error for debugging purposes and responds with a 500 Internal Server Error status and an error message: "Error occurred during sign up".

**User Redirection:**

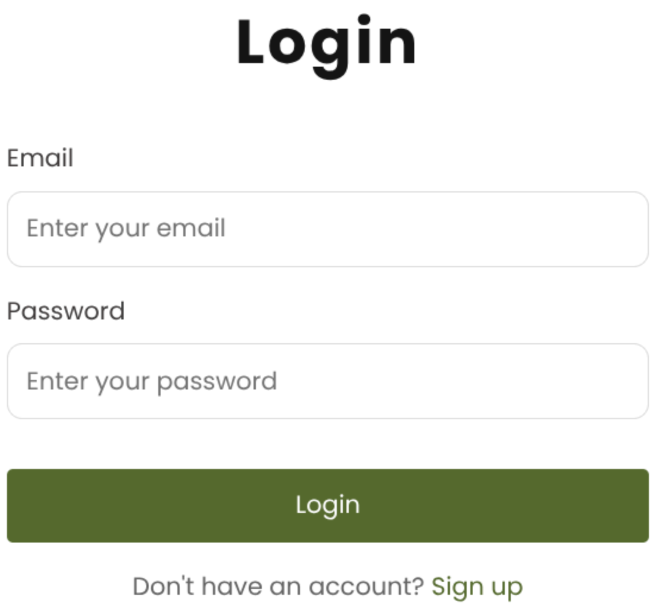
- After successfully Signing Up the frontend redirects the user to the login page , the user has to now login again to access the platform

**2. Login Workflow**

The Login process involves log in with an existing user account.

**Frontend Process**

- **User Interaction:**
    - The user navigates to the login page, where they are presented with a form,prompting the user to enter the email and password.



- **Data Submission:**
    - Once the user fills out the form and clicks the "Login" button, the frontend sends the user's credentials to the backend in the form of a JSON object via an API call. This is typically done using a POST request to the /api/v1/auth/login endpoint.

**Backend Process**

**Receiving the Request:**

- The backend receives the request at the `/api/v1/auth/login` endpoint.
- The request body contains the user's credentials (`email` and `password`), which are extracted using `req.body`.

**Input Validation:**

- The backend first checks if both required fields (`email` and `password`) are present. If any field is missing, it responds with a `400 Bad Request` status and an error message: "Email and password are required".

**Checking for User Existence:**

- The backend queries the database to check if a user with the provided email exists. If no user is found, the backend responds with a `401 Unauthorized` status and an error message: "User does not exist".

**Password Verification:**

- If the user exists, the backend compares the provided password with the hashed password stored in the database using `bcrypt.compare()`.
- If the passwords do not match, the backend responds with a `401 Unauthorized` status and an error message: `"Invalid password"`.

**Generating a JWT (JSON Web Token):**

- If the credentials are valid, the backend generates a JWT to represent the user's authenticated session. This token is signed with a secret key and contains payload information such as the user's ID and email.
- The JWT is sent back to the frontend as part of the response.

**Response to Frontend:**

**Token Handling:**

- Upon receiving a successful response from the backend, the frontend extracts the JWT (JSON Web Token) from the response.
  - The token is stored in sessionStorage to maintain the user's authenticated state across different pages of the application.
  - The response sent to the frontend looks something like this

```
  {  "message": "Login successful",
"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6Im5pZ2FtYXJlZGR5QG
dtYWlsLmNvbSIsInVzZXJJZCI6IjY3OWNkOGYyNWJiM2YyMzRjZDU2NWM3OCIsImlhdCI6MTczOD
MzOTQ0NSwiZXhwIjoxNzM4NDI1ODQ1fQ.Xx7NtA4CuN8r0M_oES3cEaUICu4Fqrd4fn5ET0x3FKA
”}
```

**User Redirection:**

- After successfully storing the token, the frontend redirects the user to the homepage
- The user is now considered authenticated, and the token is used for subsequent API requests to authorize the user.

**Image upload and processing:**

**FastAPI Backend Overview:**

- The FastAPI backend serves as the core of the application, handling image uploads, preprocessing, inference using a pre-trained ONNX model, and returning prediction results. It also communicates with a Node.js backend to submit reports.

**ONNX Model Loading:**

- The backend loads a pre-trained ONNX model using ONNX Runtime.
- The model is used for inference, and the input/output names are extracted for compatibility.

**Image Preprocessing:**

The transform pipeline preprocesses the uploaded image:

- Converts the image to grayscale.
- Resizes it to 28x28 pixels.
- Normalizes the pixel values.
- Converts the image to a tensor and adds a batch dimension.

The preprocessed image is then converted to a NumPy array for inference.

**Prediction Endpoint (/predict/):**

- This endpoint accepts an image file and optional parameters (location and description) via a POST request.

- The image is preprocessed and passed to the ONNX model for inference.
- The model outputs logits, which are converted to probabilities using a sigmoid function.
- A threshold of 0.65 is applied to determine the final prediction (e.g., pothole detected or not).
- The prediction result (label and probability) is returned to the frontend in a JSON response.

**Server Initialization:**

- The server runs on http://127.0.0.1:8000 using Uvicorn, an ASGI server for FastAPI.
- Uvicorn is a lightning-fast ASGI (Asynchronous Server Gateway Interface) server implementation for Python web applications.
- ASGI (Asynchronous Server Gateway Interface) is a standard interface between web servers and Python web applications that supports asynchronous operations.
- ASGI allows web applications to handle multiple requests concurrently, making it ideal for high-performance and real-time applications.

**React Frontend Overview:** The React frontend allows users to upload an image, submit it for prediction, and view the results. If a pothole is detected, users can provide additional details (location and description) and submit a report to the Node.js backend.

**File Upload:**

- The Upload component includes a file input field that is hidden and triggered by a button click.
- When a file is selected, its name, type, and Base64 URL are stored in the state.

**Image Preview**:

- The uploaded file is an image, it is displayed in the UI for preview.

**Prediction Submission:**

- When the user clicks "Submit," the image is sent to the FastAPI backend for prediction(/predict).

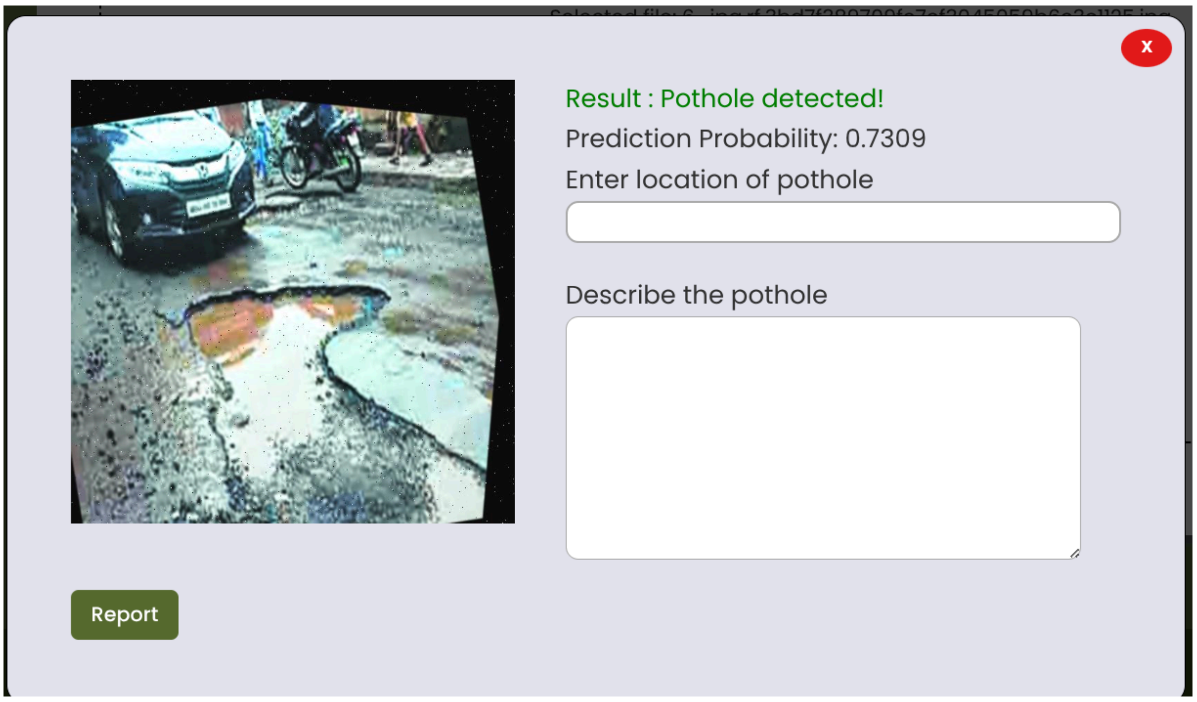
- The backend's response (prediction label and probability) is stored in the state and displayed in a popup.

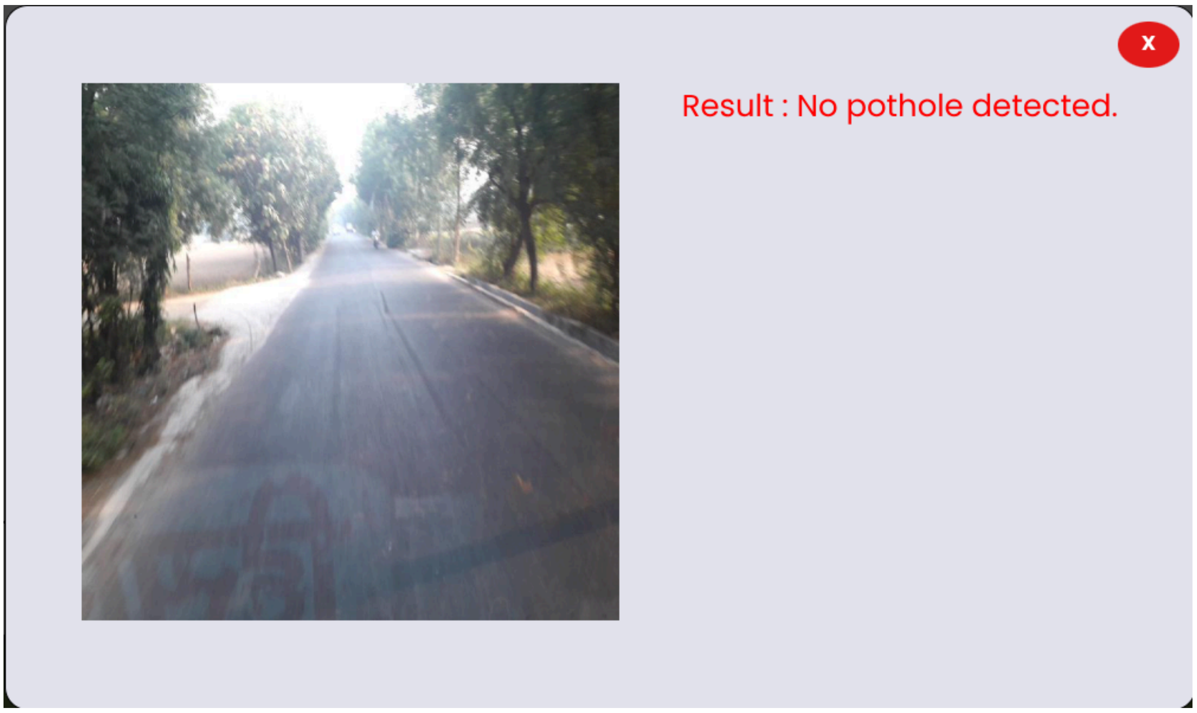
**Report Submission:**

- If a pothole is detected, the popup displays the prediction result, probability, and input fields for location and description. Users can submit a report with these details.
- The handleReportSubmit function sends the report data (image , prediction result, probability, location, and description) to a Node.js backend.
- The request includes an authorization token retrieved from sessionStorage.

**UI Components:**

- The UI includes a file upload section, a preview area, and a popup for displaying results and collecting additional information.
- Buttons for submitting the image and report are conditionally rendered based on the prediction result.


Result : Pothole detected!
Prediction Probability: 0.7309
Enter location of pothole
Describe the pothole



**Additional Features and functionality**:

**User Profile and History Management Features**

- **Get User Profile**

This feature allows authenticated users to retrieve their profile details, such as their name and email.

**Workflow:**

**Token Extraction and Verification:**

- The user's JWT token is extracted from the Authorization header of the request.
- If no token is provided, the server responds with a 401 Unauthorized error.
- The token is verified using the secret key .
- If the token is invalid or expired, the server responds with a 401 Unauthorized error.

**Fetch User Details:**

- The user's email is extracted from the decoded token payload.
- The server queries the Users collection in MongoDB to fetch the user's details (name and email).
- If the user is not found, the server responds with a 404 Not Found error.
- Response:
- The server returns the user's name and email in a JSON response.

- **Sign Out**

This feature allows users to sign out of the application. Since JWT tokens are stateless, signing out is handled on the frontend by clearing the token from storage.

**Workflow**:

- The frontend clears the JWT token from sessionStorage or localStorage.
- The backend responds with a success message indicating that the user has signed out successfully.

N
nigama reddy
nigamareddy@gmail.com
Sign Out

- **Update History**

This feature allows users to save a report of a detected pothole to their history. The report includes the image, prediction result, probability, location, and description.

**Workflow:**

**Token Extraction and Verification:**

- The user's JWT token is extracted from the Authorization header and verified using the secret  key.
- If the token is invalid or expired, the server responds with a 401 Unauthorized error.

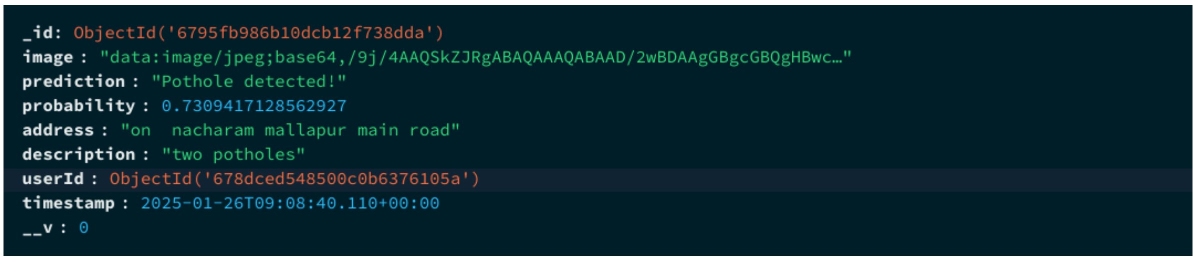
- The user ID is extracted from the decoded token payload.

**Create Report:**

- A new report document is created in the History collection, including:
    - Image (Base64 URL).
    - Prediction result (e.g., "Pothole Detected").
    - Probability (confidence score).
    - Location (address of the pothole).
    - Description (user-provided details).
    - User ID (to associate the report with the user).

**Save Report:**

- The report is saved to MongoDB.
- If successful, the server responds with a success message.



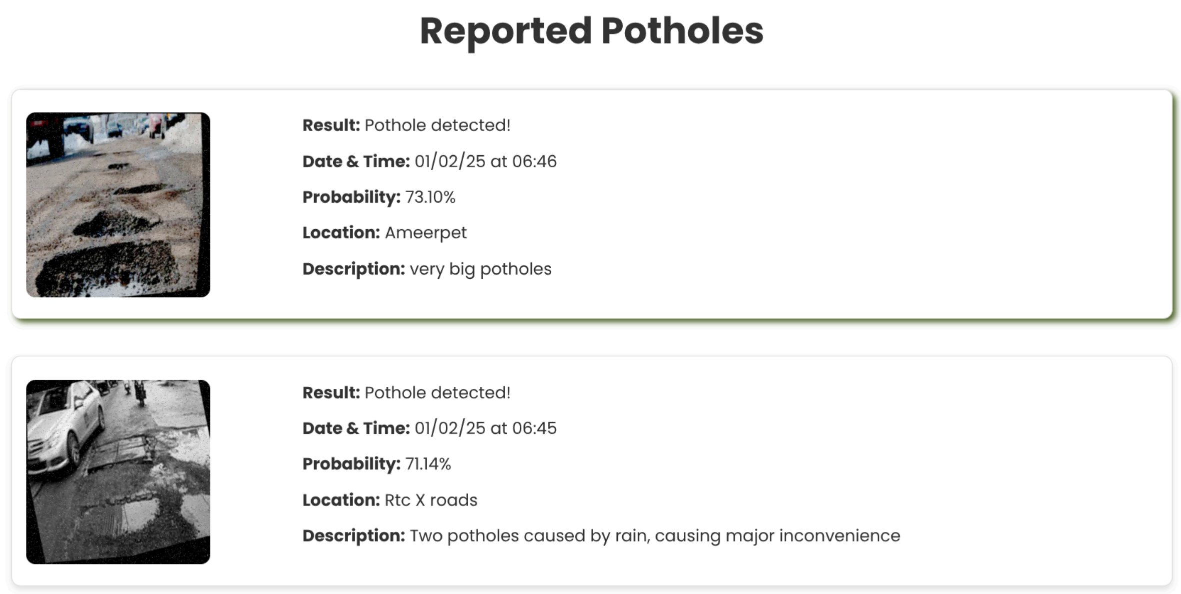
```
_id: ObjectId('6795fb986b10dcb12f738dda')
image : "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAgGBgcGBQgHBwc…"
prediction : "Pothole detected!"
probability : 0.7309417128562927
address : "on  nacharam mallapur main road"
description : "two potholes"
userId : ObjectId('678dced548500c0b6376105a')
timestamp : 2025-01-26T09:08:40.110+00:00
__v : 0
```

**Error Handling:**

- If an error occurs (e.g., database connection issue), the server responds with a 500 Internal Server Error.

- **Fetch History**

This feature allows users to retrieve their history of reported potholes, sorted by timestamp in descending order.


Reported Potholes

**Result:** Pothole detected!
**Date & Time:** 01/02/25 at 06:46
**Probability:** 73.10%
**Location:** Ameerpet
**Description:** very big potholes

**Result:** Pothole detected!
**Date & Time:** 01/02/25 at 06:45
**Probability:** 71.14%
**Location:** Rtc X roads
**Description:** Two potholes caused by rain, causing major inconvenience

**Workflow:**

**Token Extraction and Verification:**

- The user's JWT token is extracted from the Authorization header and verified using the secret key.
- If the token is invalid or expired, the server responds with a 401 Unauthorized error.
- The user ID is extracted from the decoded token payload.

**Fetch History:**

- The server queries the History collection in MongoDB to fetch all reports associated with the user ID.
- The results are sorted by timestamp in descending order (most recent first).

**Response:**

- If no reports are found, the server responds with a 404 Not Found error.
- Otherwise, the server returns the user's history as a JSON response.

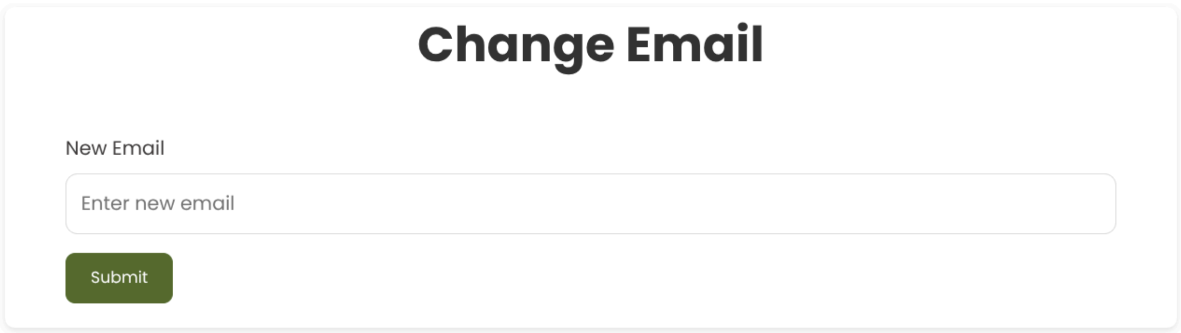
**Error Handling:**

- If an error occurs (e.g., database connection issue), the server responds with a 500 Internal Server Error.

- **Update credentials**

**Update Email :**

This feature allows users to update their email address.



**Workflow:**

**Token Extraction and Verification:**

- The user's JWT token is extracted from the Authorization header and verified using the secret  key.
- If the token is invalid or expired, the server responds with a 401 Unauthorized error.

**Validate New Email:**

- The server checks if the new email is already in use by querying the Users collection.
- If the email is already in use, the server responds with a 400 Bad Request error.

**Update Email:**

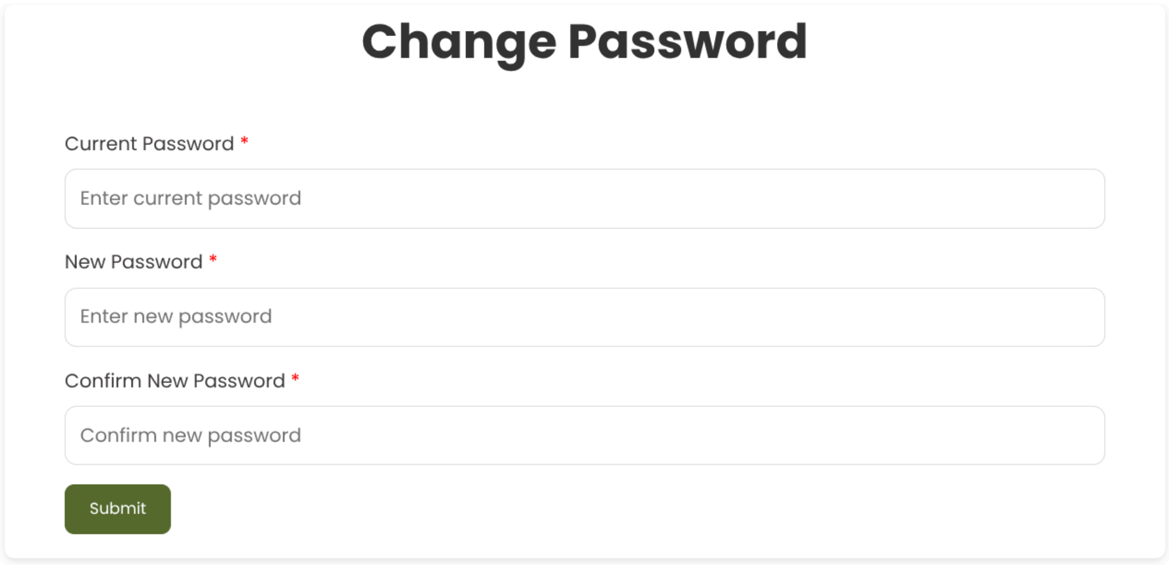
- The user's email is updated in the Users collection.
- If successful, the server responds with a success message.

**Error Handling:**

- If an error occurs (e.g., database connection issue), the server responds with a 500 Internal Server Error.

**Update Password:**

This feature allows users to update their password.


Change Password

Current Password *

Enter current password

New Password *

Enter new password

Confirm New Password *

Confirm new password

Submit

**Workflow:**

**Token Extraction and Verification:**

- The user's JWT token is extracted from the Authorization header and verified using the secret key.
- If the token is invalid or expired, the server responds with a 401 Unauthorized error.
- The server checks if all required fields (currentPassword, newPassword, confirmNewPassword) are provided.
- If the new password and confirmation do not match, the server responds with a 400 Bad Request error.

**Verify Current Password:**

- The server compares the provided currentPassword with the hashed password stored in the Users collection.
- If the current password is incorrect, the server responds with a 401 Unauthorized error.

**Update Password:**

- The new password is hashed using bcrypt and updated in the Users collection.

- If successful, the server responds with a success message.

**Error Handling:**

- If an error occurs (e.g., database connection issue), the server responds with a 500 Internal Server Error.

# VIII . CONCLUSION

We have successfully developed and demonstrated a pothole identification system that accurately and effectively detects potholes on roads using a Convolutional Neural Network (CNN) . By combining advanced image preprocessing techniques, feature extraction, and classification, our system achieves a high level of accuracy in identifying potholes. Extensive testing and evaluation have shown that our approach has a high detection rate and a low false-positive rate , making it a reliable solution for real-world applications. The integration of FastAPI and ONNX Runtime enables real-time inference, ensuring that the system can process images quickly and efficiently, even in dynamic environments. Additionally, the React-based frontend provides an intuitive and user-friendly interface, allowing users to upload images, view detection results, and submit reports seamlessly.

The system's ability to accurately detect potholes has significant implications for road safety and maintenance . By identifying potholes early and precisely, our system helps prevent accidents and reduces vehicle damage caused by poor road conditions. This not only enhances the safety of drivers and pedestrians but also contributes to a smoother and more reliable transportation network. Furthermore, the system supports cost-effective maintenance by enabling timely repairs, which can extend the lifespan of road infrastructure and reduce long-term maintenance costs. The data generated by the system, such as pothole locations and severity, provides valuable insights for transportation departments and road maintenance organizations, allowing them to prioritize repairs and allocate resources more effectively.

Despite its success, there are areas for improvement, such as enhancing the model's performance in challenging weather conditions or on roads with complex textures. However, the system's current capabilities demonstrate its potential to be a valuable tool for improving road safety and maintenance. By leveraging the power of machine learning and modern web technologies, our pothole detection system offers a practical and scalable solution for addressing one of the most common and persistent challenges in road infrastructure management. Overall, our proposed method has proven its effectiveness in accurately detecting potholes, and we believe it holds great promise for transportation departments and road maintenance organizations worldwide.

# IX. FUTURE SCOPE

Yet, there is still opportunity for advancement in the system's functionality. To further increase the accuracy and dependability of the system, more research can be done to examine the utilization of other data sources and techniques, such as data augmentation, transfer learning, and hyperparameter tuning.

In terms of future work, there is scope for implementing more models such as pedestrian detection, obstacle detection, and model to visualize the environment. This will further enhance the growth to-wards the automation of cars for the safety of the people using them.

Some other potential areas of future work are:

**Multi-object detection**: Although our technology is intended to find potholes, other road dangers like cracks or debris may also be useful to find. Future research can examine the application of multi-object detection algorithms to identify various traffic dangers.

**Integration with current systems**: To offer more comprehensive road maintenance solutions, our pothole detecting system can be linked with current systems for maintaining roads. For better and more effective road maintenance planning, the system can be connected with a road survey system, for instance.

**Generalization to other environments**: The system may be usefully generalized to other contexts, such as highways or urban roads, even if it has been trained and tested on specific road conditions and locations. Future research can examine the system's adaptability to different settings and circumstances.

# X REFERENCES

[1] https://www.ibef.org/industry/roads-presentation

[2] https://morth.nic.in/road-accident-in-india

[3] https://www.thehindu.com/opinion/op-ed/fix-the-pothole problem/article24436287.ece

[4] Dipak k das. Potholes killed 3,597 across India in 2017, terror 803.https://timesofindia.indiatimes.com/india/potholes-killed-3597 across-india-in-2017-terror-803/articleshow/64992956.cms

[5] Fundamentals of Deep learning KMIT vista https://youtu.be/XAb5yL2enkU?si=ySDGw3sUCnCwsIlE

[6] NodeJs Documentation https://nodejs.org/docs/latest/api/

[7] Express Documentation https://expressjs.com/en/5x/api.html

[8] Fast API Docs https://fastapi.tiangolo.com/reference/

[9] React Docs https://react.dev/reference/react

[10] MongoDB Docs https://www.mongodb.com/docs/