A Project for the Degree of Bachelor of Science in Computer and Information Technology

Object Classification Using Light Weight CNN

Submitted for the partial fulfillment of the requirement for the degree of Bachelor of Science in Computer and Information Technology

Anita Banjara (T.U Exam Roll No:21234/075)

Ashmita Basnet (T.U Exam Roll No:21239/075)

Badri Raj Aran (T.U Exam Roll No:21240/0753)

Swastik College

Department of Computer Science and Information Technology

Institute of Science and Technology

Tribhuvan University

April, 2023

Supervisor's Recommendation

I hereby recommend that the project has been prepared under my supervision by the team of Anita Banjara, Ashmita Basnet and Badri Raj Aran entitled **" Object Classification Using Light Weight CNN "** in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology be processed for evaluation.

.....

Mr. Mohan Bhandari

Supervisor

Lecturer

Swastik College

Certificate for Approval

This is to certify that this project prepared by Anita Banjara, Ashmita Basnet and Badri Raj Aran entitled **"Object Classification Using Light Weight CNN"** in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

Mr.Mohan Bhandari
Supervisor
Swastik College

.....
External Examiner

.....
Mrs. Shristi Khatiwada
Head of Department
Swastik College

Abstract

Object Detection has been one of the areas of interest of research community for over years and has made significant advances in its journey so far. Achieving new heights in objects detection and image classification was made possible because of Convolution Neural Network (CNN). In the current work, we have implemented real time object detection and have made efforts to improve the accuracy of the detection mechanism. Without compromising the classification accuracy and better feature extraction, deep learning (DL) model. Hence cifar-10 data set is provided in this algorithm designed platform and then machine is trained accordingly.
The project focuses on developing a high-performance prediction model using the latest techniques and tools in deep learning. Here we use cifar10 datasets. There are a total of two convolutional layers and two max pooling layers in this CNN architecture. With CNN, at the end 100 epochs, accuracy was at around 98 percentage with an average processing time of 48ms/step.

Keywords: Object Detection, Convolutional neural network (CNN), Cifar10 Dataset, Activation Function, image processing.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
CNN	Convolution Neural Network
DL	Deep Learning
ML	Machine Learning
OD	Object Detection

Chapter 1 Introduction

1.1 Introduction

In the recent years, there has been an exponential progress in the field of machine learning and artificial intelligence which has led to improvement in accuracy, reduction in human efforts and failure rate. This development has played a commendable role in reducing processing time, which has further led to improvement in net productivity and corresponding reduction in the cost.

Object Detection(OD) is a challenging and exciting task in computer vision.It can be difficult since there are all kinds of variations in orientation, lighting, background and occlusion that can result in completely different images of the very same object.

The widely used object detection applications are human–computer interaction, video surveillance, satellite imagery, transport system, and activity recognition. In the wider family of deep learning architectures, convolutional neural network (CNN) made up with set of neural network layers is used for visual imagery. Deep CNN architectures exhibit impressive results for detection of objects in digital image.

The project comes with the technique of **"Object Classification Using Light Weight CNN"** which includes various research sides of computer science. The project is to take a picture of an object and process it up to recognize the image of that object like a human brain recognize the various objects[1]. The project contains the deep idea of the Image Processing techniques and the big research area of machine learning(ML) and the building block of the machine learning called Neural Network .

1.2 Problem Statement

Object detection is a bit more complex task. We donot have proper algorithm to create the automated system and real time object detection. So through this project different real time objects can be easily detected even in the low resolution and develop a machine learning model that can predict the object present in an image in real-time using a lightweight convolutional neural network (CNN) architecture. Using CNN, it resolves problems like overfitting and also reduces time complexity. It can have input as images in proper grayscale size and then the images are fragmented in smaller sizes for better clearity and understanding of machine.

1.3 Project Objectives

The main objective of this project is to design and implement a CNN that can accurately predict objects in real-time using the CIFAR10 dataset . The specific goals of the project are:

1. To train the model on the CIFAR10 dataset and evaluate its performance in terms of accuracy and speed.

2. To optimize the model to reduce the response time and improve the prediction accuracy, making it suitable for various real-world applications.

1.4 Scope and Limitations

The scope of the "Object Classification Using Light Weight CNN" project encompasses the following areas:

1. The project aims to develop a CNN model that can accurately predict objects in real-time with a fast response time.

2. The model performance will be evaluated in terms of accuracy and response time, and further optimized to improve its performance.

The "Object Classification Using Light Weight CNN" project may face the following limitations:

1. More than 10 objects cannot be detected.

2. The model may not perform well on real-world objects and scenes that are significantly different from those in the CIFAR10 dataset, leading to lower accuracy and prediction errors.

3. The model may not generalize well to other datasets and real-world scenarios, requiring further fine-tuning or transfer learning techniques to adapt to new environments.

1.4.1 Development Methodology

The model chosen for this system is “WaterFall model”. This is a simple project with the well-defined process and the requirement. The various steps of the waterfall model is followed throughout the whole project. This model is understood and incorporate in this project.Very less user is involved during the development of these projects. Thus, this system is developed

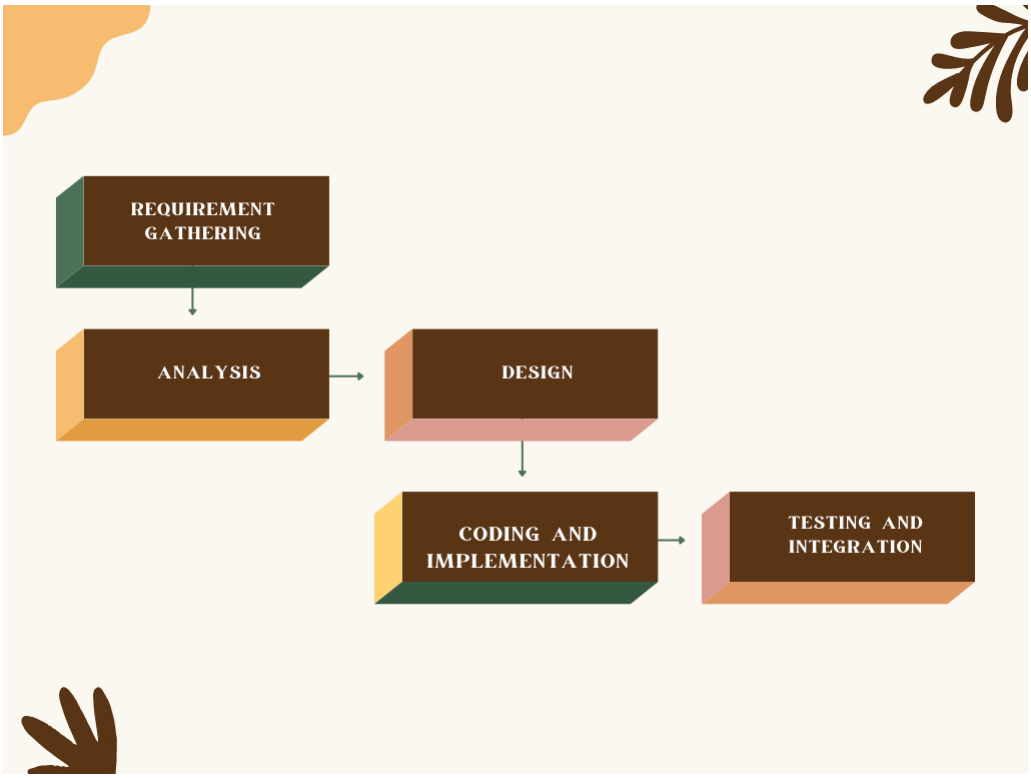


Figure 1.1: Waterfall Model

according to the Waterfall model. The various Steps of the Waterfall is shown in figure 1.1 The "Object Classification Using Light Weight CNN" project can be developed using the waterfall model, which is a sequential and linear approach to software development. The following are the main phases of the waterfall model that can be applied to this project:

1. **Requirements gathering:**

   In this phase, the requirements and specifications for the project are gathered and documented.We identify the specific datasets required for this project, including the CIFAR10 dataset required to train and test the CNN model.The training dataset consists of 50,000 32X32 colour images of 10 different objects. These are the different classes of images in the dataset which consist of aeroplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck.

2. **Analysis and design:**

   In this phase, the requirements are analyzed and a detailed design of the model and user interface is developed.We trained the CNN model on the Cifar10 dataset using a machine learning development tool called Google Colaboratory and define the user interface .

3. **Coding and Implementation:**

   In coding phase, system backend is developed by using python. After this the model and

user interface are implemented and developed. We trained the model on the CIFAR10 dataset, built the user interface and integrate the model with the user interface.

4. **Testing:**

   In this phase, the model and user interface are tested and validated to ensure they meet the specifications and requirements. With CNN, at the end 100 epochs, accuracy was at around 98 percent with an average processing time of 48ms/step.

1.4.2 Report Organization

The report is separated into different chapters which are as follows:
The report organization gives the information about division of different section in the project. The report is started with Chapter 1 that explains about introduction, problem statement, objectives, scope and limitations . Chapter 2 explains about the literature review related to the paper research for the system development. Chapter 3 discussed about the system analysis where requirement analysis and feasibility analysis are required in system development. Chapter 4 deals about the system design and modelling part of the development process. In, Chapter 5 we discussed about the implementation and testing of the system. At last, Chapter 6 conclude the system development .

Chapter 2 Background Study and Literature Review

2.1 Background

2.1.1 Deep learning

Deep learning, a subset of machine learning which in turn is a subset of artificial intelligence (AI) has networks capable of learning things from the data that is unstructured or unlabeled. The approach utilized in this project is Convolutional Neural Networks (CNN). Deep learning is a popular technique used in computer vision[2]. We chose Convolutional Neural Network (CNN) layers as building blocks to create our model architecture. CNNs are known to imitate how the human brain works when analyzing visuals [3]. A typical architecture of a convolutional neural network contains an input layer, some convolutional layers, some dense layers (aka. fully-connected layers), and an output layer. These are linearly stacked layers ordered in sequence.

2.1.2 Convolution Neural Network(CNN)

CNN is a deep learning algorithm that take an input image, assign learnable biases and weights to numerous objects in the image and differentiate one from the other. As comparing to other classification algorithm, the preprocessing needed in CNN is much lower and are used in object detection, plant disease detection, fraud identification and many more.It is like the connectivity of neurons pattern in human brain and motivated by the visual cortex organization. A CNN model works in three stages. In the first stage, a convolutional layer extracts the features of the image/data. In the second stage a pooling layer reduces the dimensionality of the image, so small changes do not create a big change on the model. Simply saying, it prevents over fitting. In the third stage a flattening layer transforms our model in one-dimension and feeds it to the fully connected dense layer. This dense layer then performs prediction of image. A good model has multiple layers of convolutional layers and pooling layers.

2.1.3 Activation functions

Activation functions are used in the layers of the Convolutional Neural Network (CNN) model to introduce non-linearity into the model. This non-linearity allows the model to learn complex representations of the input data, which is important for accurate object recognition and prediction.There are several activation functions that can be used in the layers of a CNN model.ReLU

is used in our project as an activation function.

1. ReLU (Rectified Linear Unit)

   ReLU is a popular activation function that replaces negative values with zero and leaves positive values unchanged [4]. This activation function is computationally efficient and has been shown to improve the convergence of the model.ReLU is an important component in the "Object Classification Using CNN". It helps to improve the accuracy of predictions, speed up computation time, and provide a non-linear decision boundary for the model to learn from.

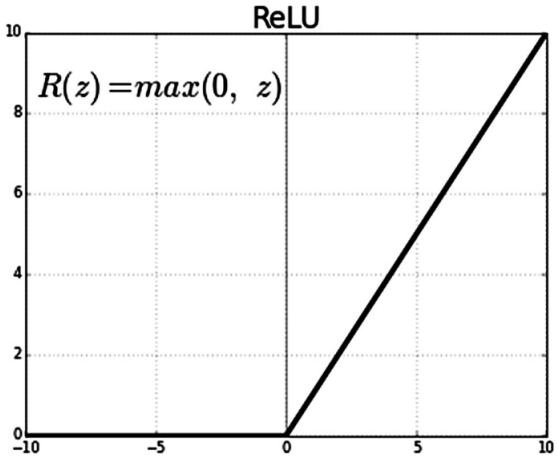


Figure 2.1: ReLU Graph

2.2 Literature Review

A literature review of the topic "Object Classification Using CNN" involves exploring the existing research and studies related to each of the components mentioned in the title.

G. Yao, T. Lei, and J. Zhong [1], “A review of convolutional-neural-network-based action recognition,”in their article describe convolutional neural network (CNN) is one of the most popular and used of DL networks. Because of CNN, DL is very popular nowadays. The main advantage of CNN compared to its predecessors is that it automatically detects the significant features without any human supervision which made it the most used. Therefore, we have dug in deep with CNN by presenting the main components of it.

In their paper,S. S. Basha, S. R. Dubey, V. Pulabaigari, and S. Mukherjee [5], have conducted extensive experiments to observe the useful practices in deep learning for the usage of Convolutional Neural Networks (CNNs) .The four CNN models are implemented to perform the experiments on publicly available CIFAR-10 dataset.In thir paper, they have analyzed the effect of certain decisions in terms of the FC layers of CNN for image classification. Careful

selection of these decisions not only improves the performance of the CNN models but also reduces the time required to choose among different architectures such as deeper and shallow.
Doon, T. Kumar Rawat, and S. Gautam[6],in their project used a deep neural network to perform image classification on CIFAR-10 dataset. ReLU activation function is used throughout the network except at the output layer where Softmax activation function is used. The output labels are in the form of integers and need to be changed to categorical using one hot encoding. During training each forward pass results in an output value which is compared with the actual output and loss function is computed . Cross-entropy loss function is used along with Softmax activation at the output and the gradient of the loss function are propagated back into the network to optimize the weights. This process is repeated for fixed number of epochs to get best possible accuracy.
In [7], Training the deep learning models involves learning of the parameters to meet the objective function. Typically the objective is to minimize the loss incurred during the learning process. In a supervised mode of learning, a model is given the data samples and their respective outcomes. When a model generates an output, it compares it with the desired output and then takes the difference of generated and desired outputs and then attempts to bring the generated output close to the desired output. This is achieved through optimization algorithms. An optimization algorithm goes through several cycles until convergence to improve the accuracy of the model. There are several types of optimization methods developed to address the challenges associated with the learning process. Six of these have been taken up to be examined in this study to gain insights about their intricacies. The methods investigated are stochastic gradient descent, nesterov momentum, rmsprop, adam, adagrad, adadelta. Four datasets have been selected to perform the experiments which are mnist, fashionmnist, cifar10 and cifar100. The optimal training results obtained for mnist is 1.00 with RMSProp and adam at epoch 200, fashionmnist is 1.00 with rmsprop and adam at epoch 400, cifar10 is 1.00 with rmsprop at epoch 200, cifar100 is 1.00 with adam at epoch 100. The highest testing results are achieved with adam for mnist, fashionmnist, cifar10 and cifar100 are 0.9826, 0.9853, 0.9855, 0.9842 respectively. The analysis of results shows that adam optimization algorithm performs better than others at testing phase and rmsprop and adam at training phase.

Chapter 3 System Analysis

3.1 System Analysis

System Analysis is one of the important phases of system development. After the planning phase, a system analysis is done. Planning is the phases in which we decide what kind of the system we are going to developed. In Analysis phase different analysis is done such as Requirement Analysis, Feasibility Analysis and so on. This phase decides whether a planned system would be feasible or not. Whether these system meets the requirement of the user or not. Here, we have done Requirement analysis and Feasibility Analysis. System analysis involves Requirement analysis(Functional Requirements , Non-Functional Requirements) and Feasibility Analysis.

3.1.1 Requirement Analysis

Requirement analysis is one of the most important phases during system development.Requirement analysis is significant and essential activity after elicitation.It consists of the functional and non-functional requirement.

1. **Functional Requirements** A functional requirement document defines the functionality of a system.They are basically the requirements stated by the user which one can see directly in the final product. The system can be used to classify images into different categories, such as animals, objects, and scenes. The use case diagram is shown in Figure 3.1.

   (a) Actor:

   The user or client who wants to predict the objects in real-time.

   (b) Model:

   A light-weight convolutional neural network (CNN) model that has been trained on the Cifar10 dataset. The model takes an image as input and predicts the class of the object in the image.

   (c) Webpage:

   A webpage that allows the user to upload an image and get a prediction of the object in the image. The webpage is designed using a framework tenserflow and

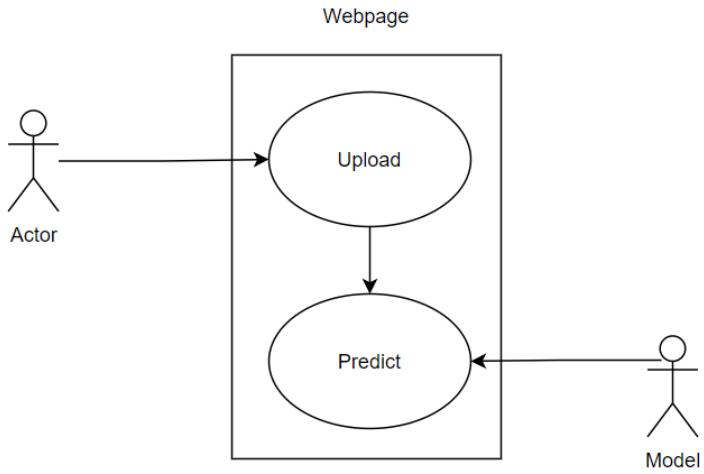


Figure 3.1: Use Case Diagram

flask. The flask is used to build the user interface and handle user interactions, such as the image upload.

(d) Upload:

The user uploads an image file through the webpage. The image is uploaded that sends the file to the backend server for processing.

(e) Predict:

Once the image is uploaded, the model processes the image and returns a prediction of the object in the image. The prediction is displayed on the webpage to update the UI with the prediction result.

2. **Non-Functional Requirements**

Non-functional requirement is those which doesn't directly deals with the function of the system.These are basically the quality constraints that the system must satisfy according to the project contract. Non-functional Requirement define system attributes such as security, reliability, performance, maintainability, scalability, and usability. Some of the nonfunctional requirements of this system are:

(a) **Prediction:**

The CNN model was trained on the Cifar10 dataset, which contains images of 10 different object classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). To make predictions, the CNN was process an input image and generate a score for each of the 10 classes, indicating the likelihood that the image

belongs to each class. The class with the highest score was the prediction output of the model. The prediction process should be fast enough to meet the requirement of real-time object recognition.

(b) **Accuracy**

To measure the accuracy of the model, we split the Cifar10 dataset into a training set and a validation set. The model was trained on the training set, and the accuracy was evaluated on the validation set. The accuracy is reported as a percentage of correctly classified objects.

3.1.2 Feasibility Analysis

Feasibility analysis is done to check whether the development of this system is feasible in nature or not. Feasibility analysis is very essential process in system development.Feasibility analysis helps to determine whether development of these system is feasible or not. Feasibility analysis is necessary to map out all details like cost, income, time so on.During this project we have done.

1. **Technical Feasibility**

   Technical feasibility involves the evaluation of the hardware, software, and other technical requirements of the proposed system. The software and tools require for this system is easily available in web. Thus, we can say this system is a technically feasible.

2. **Schedule feasibility**

   This type of feasibility analysis is done to determine whether the proposed system can be developed within the given time or not. We worked with schedule as shown 3.2 throughout whole project.

Activities	Month	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9
Planning	Jan	■								
Requirement Analysis	Jan		■	■						
System Analysis	Feb				■					
Design	Feb					■	■			
Coding and Implementation	Feb						■	■		
Testing	April							■	■	
Documentation	April								■	■

Figure 3.2: Gantt Chart of Scheduled Feasibility

3. **Operational Feasibility**

   Operational feasibility is the ability to utilize, support and perform the necessary tasks of a system or program. This system is operationally feasible because it can predict whether the given cifar10 object and it is also useability, reliability.

4. **Economic Feasibility**

   The proposed system is economically feasible as it does not require enormous amount of money to be developed. The system will deliver fast and effective automated environment instead of slow and error prone manual system, thus reducing both time and manpower spent in running the system. And the system will have GUI interface and very less user-training is necessary to learn it.

3.2 Dataset Description

The CIFAR-10 dataset (Canadian Institute for Advanced Research) is a collection of images that are commonly used to train machine learning and computer vision algorithms.
In this project, a CNN model is trained on the CIFAR10 dataset, which is a widely used dataset for object recognition and image classification tasks. The CIFAR10 dataset contains 60,000 32x32 color training images and 10,000 test images, covering 10 classes of objects, such as airplanes, cars, birds, and cats[8].
The training process involves using the input data (the images from the CIFAR10 dataset)[9] to adjust the weights and biases of the model so that it can accurately recognize and predict the class of an object in an image.
Once the model is trained, it can be used for real-time object recognition by processing an input image and making a prediction based on the learned representations. The user interface is used to allow users to input images and receive the predictions made by the model.
Here are the classes in the dataset , as well as 10 random images from each shown in 3.3: The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

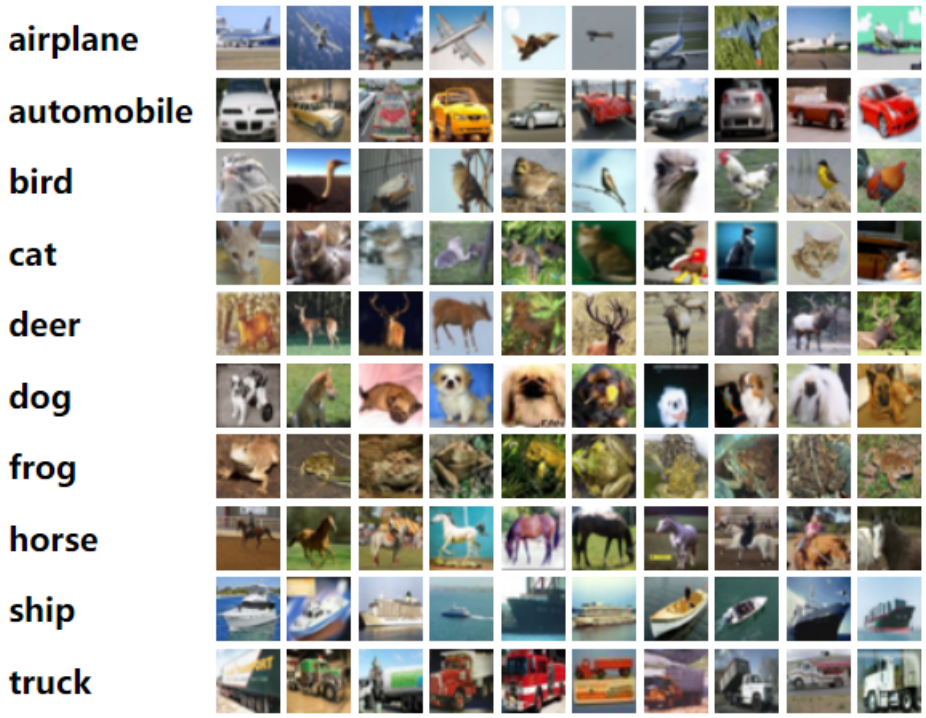


Figure 3.3: Cifar10 DataSets

3.3 Analysis

3.3.1 Activity Diagram

An activity diagram is a type of flowchart that shows the flow of activities or actions within a system. In the context of a "Object Classification Using Light Weight CNN" project, figure 3.4 is used to visualize the flow of actions involved in making a prediction using the system.

1. **Build Model:**
   This step involves building a light-weight CNN model using the Cifar10 dataset. The model is trained using deep learning techniques to recognize the 10 different objects in the dataset.

2. **Upload Image:**
   In this step the image of an object is uploaded to recognize using Flask. The image is uploaded to the server for processing.

3. **Prediction:**
   Once the image is uploaded, the trained model is used to make a prediction of the object in the image. The prediction is made on the server using the uploaded image as input.

4. **Output:**

The prediction result is displayed to the user.

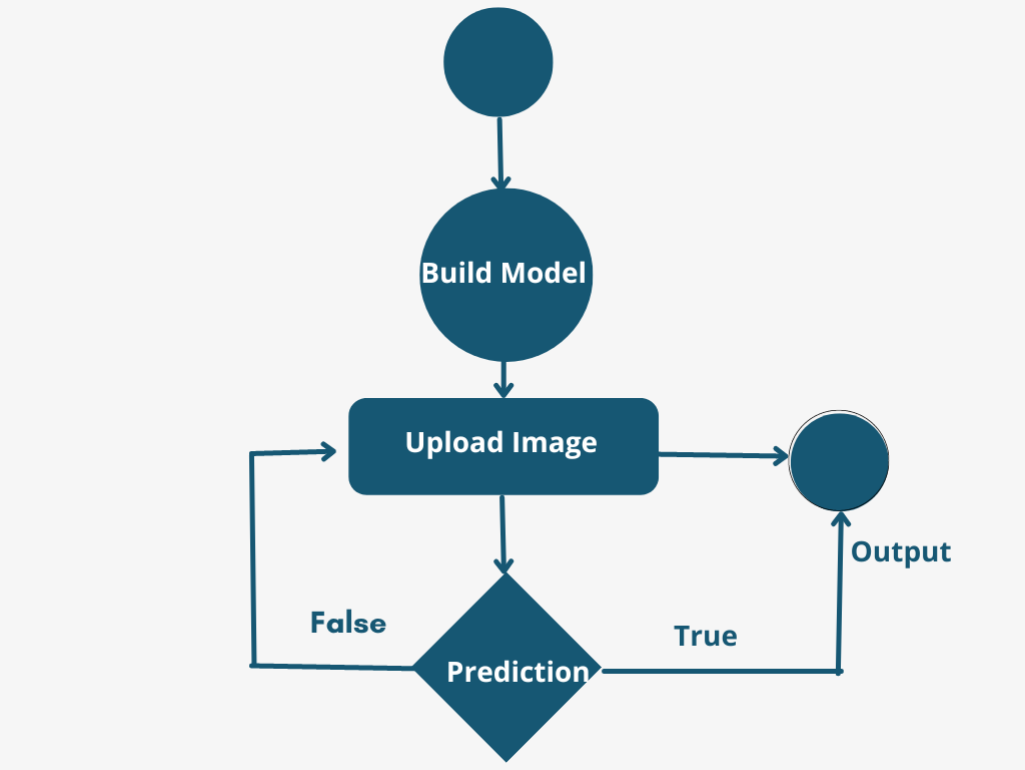


Figure 3.4: Activity Diagram

Chapter 4 System Design

4.1 CNN Configuration

In simpler words, CNN is an artificial neural network that specializes in picking out or detect patterns and make sense of them. Thus, CNN has been most useful for image classification. A CNN model has various types of filters of different sizes and numbers. These filters are essentially what helps us in detecting the pattern.

The convolutional neural network, or CNN for short, is a specialized type of neural network model designed for working with two-dimensional image data, although they can be used with one-dimensional and three-dimensional data. Central to the convolutional neural network is the convolutional layer that gives the network its name. This layer performs an operation called a "convolution". A CNN model generally consists of convolutional and pooling layers. It works better for data that are represented as grid structures, this is the reason why CNN works well for image classification problems.

The dropout layer is used to deactivate some of the neurons and while training, it reduces over fitting of the model. Our model is composed of feature extraction with convolution and binary classification. Convolution and max pooling are carried out to extract the features in the image, and a 32 3x3 convolution filters are applied to a 28x28 image followed by a max-pooling layer of 2x2 pooling size followed by another convolution layer with 64 3x3 filters. A detailed visual explanation is shown in Figure 4.1. The typical structure of a CNN consists of three basic

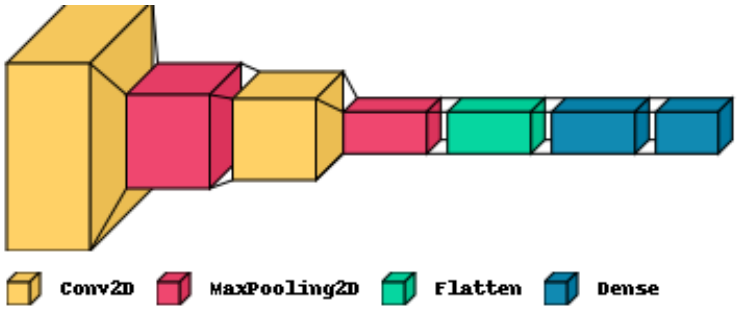


Figure 4.1: Detail architecture of CNN model

layers:

1. **Convolutional Layer:**

   Convolutional layers are the major building blocks used in convolutional neural networks. A convolution is the simple application of a filter to an input that results in an

activation[10]. We have used sequential model and used 2D convolution with 3,3 kernel with 32 filters. MaxPooling method is used for downsizing the dimensions with relu activation function.We have used output layer with 'softmax' activation function.Model is again compiled with optimizer 'adam'.The result is highly specific features that can be detected anywhere on input images.We tested our model with different kernel sizes, including 1x1, 3x3, and 5x5. We found that the model achieved the best accuracy with a 3x3 kernel size.

The numerical results of the network on the CIFAR-10 dataset are reported. We achieved an accuracy of 98 percent on the test set, which is a good performance for this dataset.

2. **Pooling Layer:**

Pooling layers are one of the building blocks of Convolutional Neural Networks. Where Convolutional layers extract features from images.

In this project we used a max-pooling layer with a size of 2x2 and a stride of 2 after the first convolutional layer. This layer reduces the spatial size of the feature maps by a factor of 2 in both dimensions.

To reduce the dimensions of the hidden layer by combining the outputs of neuron clusters at the previous layer into a single neuron in the next layer.

There are two types of pooling that are used:

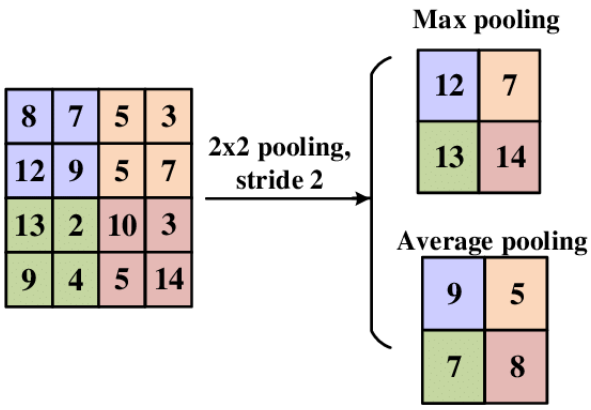


Figure 4.2: Types of Pooling Layer

(a) **Max pooling:**

This works by selecting the maximum value from every pool. Max Pooling retains the most prominent features of the feature map, and the returned image is sharper than the original image.

(b) **Average pooling:** This pooling layer works by getting the average of the pool. Average pooling retains the average values of features of the feature map. It smooths the image while keeping the essence of the feature in an image.

3. **Fully Connected Layer**

   A fully connected layer refers to a neural network in which each neuron applies a linear transformation to the input vector through a weights matrix. As a result, all possible connections layer-to-layer are present, meaning every input of the input vector influences every output of the output vector.

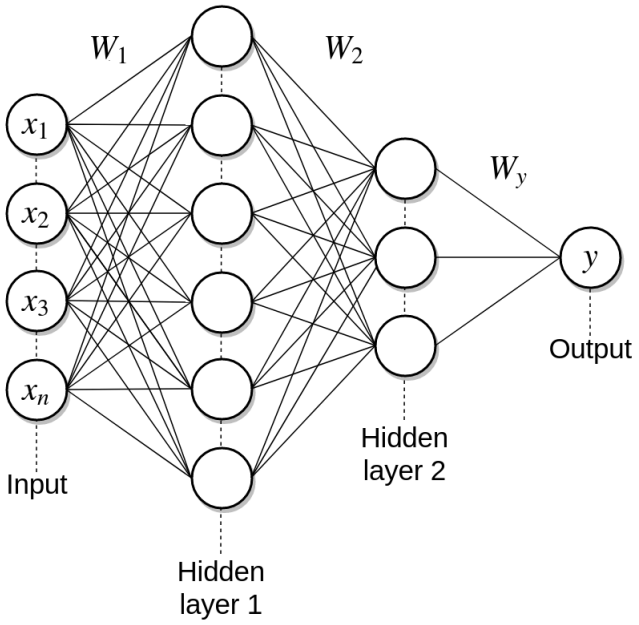


Figure 4.3: Fully Connected Layer

Summary of the model is shown in the table 4.1

Model: "Sequential"

Layer(Type)	Output Shape	Param #
Conv2d(Conv2)	(None, 30, 30, 32)	896
max_pooling2d(Max_pooling2D)	(None, 15, 15, 32)	0
conv2d_1(Conv2D)	(None, 13, 13, 64)	18496
max_pooling2d_1(MaxPooling2D)	(None, 6, 6, 64)	0
flatten(Flatten)	(None, 2304)	0
dense(Dense)	(None, 64)	147528
dense_1(Dense)	(None, 10)	650
Total params: 167,562 Trainable params: 167,562 Non-tainable: 0		

Table 4.1: CNN Details

4.2 System Workflow

Figure 4.4 shows the block diagram for "Object Classification Using light weight CNN".

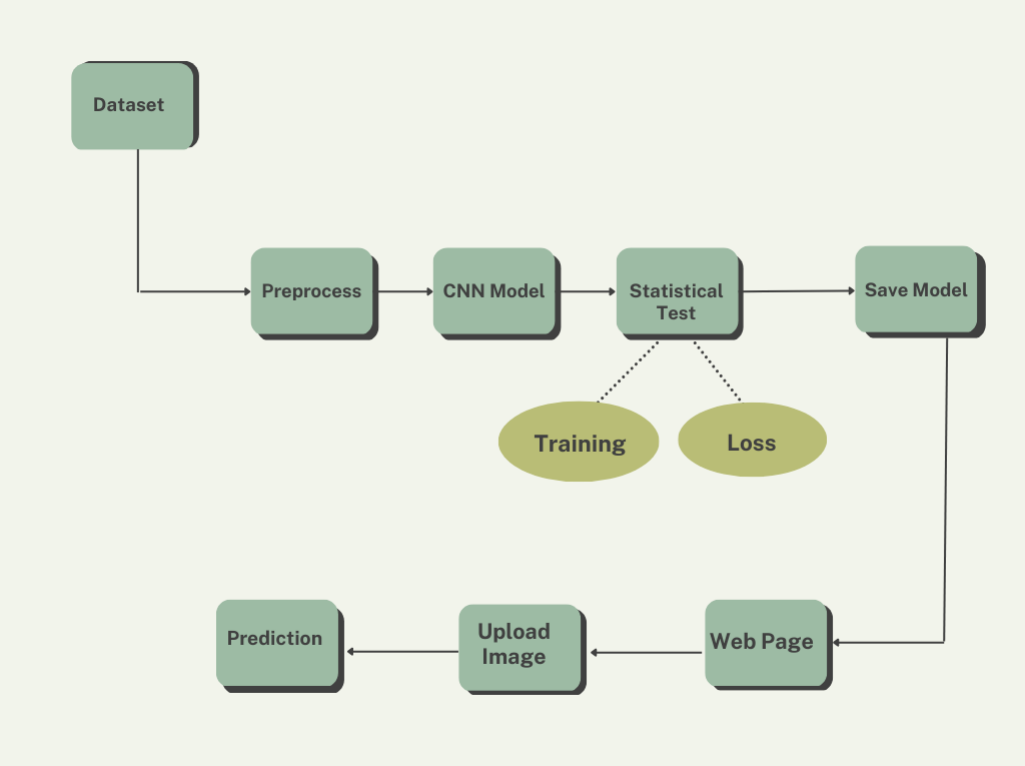


Figure 4.4: Block Diagram of Object Detection System

1. **Input:** A real-time image is fed into the system as input.

2. **Pre-processing:** The input image is pre-processed to normalize the pixel values and resize the image to a standard size suitable for the model.

3. **Convolutional Neural Network (CNN):** The pre-processed image is then fed into the Light Weight CNN model, which consists of several layers of Convolution, Pooling, and fully connected layers. The CNN extracts relevant features from the image and reduces the dimensionality of the data.

4. **CIFAR-10 Datasets:** The CNN is trained on the CIFAR-10 datasets, which consists of 10 classes of objects, such as airplanes, cars, birds, etc. The model learns the features and relationships between the image and the object class during training.

5. **Stastical Test:**The model is trained and validated by the process of visualization of history of fitting. Model is trained using training data and validation data with 100 epochs.To avoid the overfitting problem, we need to expand artificially our dataset.

6. **Save Model:**After training the model, save the model into .h5 file and deploy the model to the web which include converting the .h5 extension file and deploy the js model to the web.

7. **UI:** The predicted class label and probability score are then displayed on the web interface built using Flask, updating the UI in real-time as new images are fed into the system.

8. **Upload Image:**The image is uploaded in the system.

9. **Prediction:** The model makes predictions on the input image, assigning a class label with a probability score to the object in the image.

10. **Output:** The predicted class label and probability score are the output of the system, which can be viewed through the web interface.

4.3 Modular Decomposition

An "Object Classification Using CNN" project involves developing a web-based application that uses a light weight Convolutional Neural Network (CNN) to perform real-time object recognition. The application is built using Flask for building user interfaces, and is connected to the CNN model that is trained on the Cifar10 dataset. We know that overfitting is generaly

occur when we don't have enought data for training the model. To avoid the overfitting problem, we need to expand artificially our dataset. The idea is to alter the training data with small transformations to reproduce the variations occuring when someone is writing a digit.We have trained and validate the process by the visualization of history of fitting. We have used sequential model and used 2D convolution with 3,3 kernel with 32 filters.At the end 100 epochs, accuracy was at around 98 percentage which is a significant improvement over ANN. CNN's are best for image classification and gives superb accuracy. Also computation is much less compared to simple ANN as maxpooling reduces the image dimensions while still preserving.

4.4 Component Level Design

Sequence Diagram :

To visualize how the operations are carried out in the system, sequence diagrams are created.The vertical axis of the diagram is used to represent the time, what messages are sent and when. It helps to plan and understand the detail functionality of the developed or developing system. It is also known as event diagram. Figure 4.5 shows the sequence diagram.

Start: The user starts the application to perform object classification using CNN.

Input: The user provides an image as input to the system for classification.

Preprocessing: The input image is preprocessed to prepare it for classification. This could involve steps such as resizing, normalization, or data augmentation.

Forward Pass: The preprocessed image is passed through the CNN model in a forward pass. The model computes the output probabilities for each class based on the learned weights and biases.

Prediction: The class with the highest probability is selected as the predicted class for the input image.

Output: The predicted class label is displayed to the user as output.

End: The application ends.

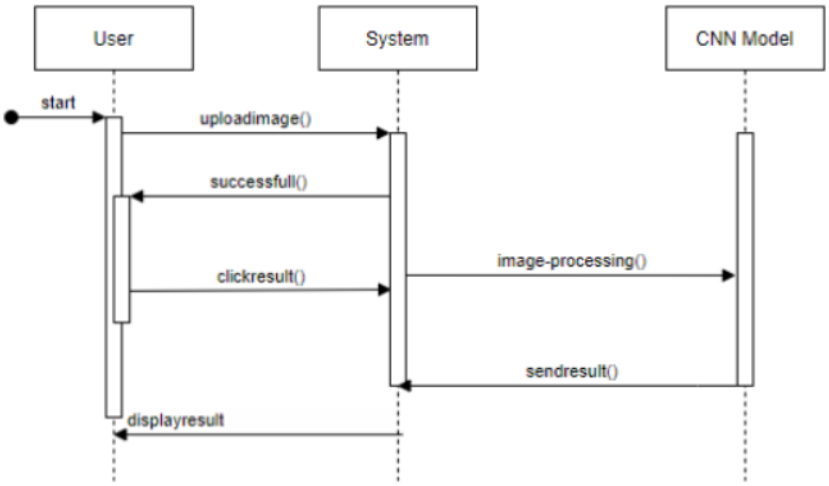


Figure 4.5: Sequence Diagram

4.5 Algorithm Details

In the project the CNN architecture consists of two convolutional layers, each followed by a max pooling layer, and two dense layers at the end. The input data has shape (32,32,3), where 32x32 represents the size of the image and 3 represents the number of color channels (RGB). Mathematically, the convolution operation can be expressed as:

$$Z[i,j] = (W * X)[i,j] + b \tag{4.1}$$

Where Z is the output feature map, W is the filter matrix, X is the input image, b is a bias term, and * denotes the convolution operation. The filter matrix W is learned during training using backpropagation, and the bias term b is a scalar value that is added to each element of the output feature map.

The activation function used in this layer is 'relu' (Rectified Linear Unit), which introduces non-linearity to the model. The relu activation function is defined as

$$f(x) = max(0, x) \tag{4.2}$$

which means that if the input to the activation function is negative, it will output 0, and if it is positive, it will output the same value.

The 'relu' activation function is used after this convolutional layer. Another max pooling layer with a pool size of (2,2) is then applied, resulting in an output with dimensions (6,6,64).

After the second pooling layer, a flatten layer is applied to convert the 3D output into a 1D

vector. This 1D vector is then passed through two dense layers. The first dense layer has 64 neurons with 'relu' activation, and the second dense layer has 10 neurons with 'softmax' activation. The softmax function is used to produce a probability distribution over the 10 possible classes in the CIFAR-10 dataset.

The CNN is compiled using the 'adam' optimizer, loss function, and 'accuracy' metric. During training, the model is trained for 100 epochs, and the training data is divided into batches of 32 samples each.

During the training process, the CNN learns to adjust the values of the weights in each layer so as to minimize the loss function. The optimization algorithm (in this case, Adam) uses backpropagation to calculate the gradients of the loss function with respect to the weights, and updates the weights accordingly.

With CNN, at the end 100 epochs, accuracy was at around 98 percentage with an average processing time of 48ms/step. CNN's are best for image classification and gives superb accuracy. Also computation is much less compared to simple ANN as maxpooling reduces the image dimensions while still preserving the features.

Chapter 5 Implementation and Testing

5.1 Implementation

The system implementation of the project "Object Classification Using Light Weight CNN" involves several components.

First, the lightweight CNN model is trained on the Cifar10 dataset using a machine learning development tool called Google Colaboratory. Once the model is trained, it can be exported and integrated into the system.

Next, a real-time object detection component is implemented using the model to classify objects in images.

Finally, a web-based user interface is developed to allow users to interact with the real-time object detection component. The user interface include features a display of the detected objects, and user controls for adjusting the settings of the real-time object detection component.

Overall, the system implementation of this project involves integrating the trained CNN model into a real-time object detection component and developing a user interface to allow users to interact with the system in real-time using a web browser.

5.1.1 Implementation Tools

1. **Hardware Requirements**

   All the internet connected desktops/laptops.

2. **Software Requirements**

   Jupyter Notebook:

   Jupyter notebooks are a popular tool used by data scientists, machine learning engineers, and researchers to create and share code, visualizations, and narrative text. They allow you to write and execute code in your web browser and mix it with formatted text, equations, and visualizations. Jupyter notebooks support a wide range of programming languages, including Python, R, Julia, and many others.

   Google Colab is a free Jupyter notebook that allows to run Python in the browser without the need for complex configuration.

   Google Collaboratory:

   Google Colab (short for "Google Colaboratory") is a web-based platform that allows users to run Jupyter notebooks in a cloud-based environment. It is a free service provided

by Google that enables users to write and execute Python code in their web browser, without having to install any software or configure any environment on their local machine.

In the project "Object Classification Using Light Weight CNNS", the Google Collab is used to create, train and test the machine learning model using the CIFAR-10 dataset. The lightweight CNN model is implemented using a deep learning framework such as TensorFlow. The collab also contained the code to preprocess the dataset, visualize the model's performance.

3. **Library**

   Tensor Flow:

   TensorFlow is a popular deep learning framework used for building and training machine learning models. In particular,TensorFlow has extensive support for building and training convolutional neural networks (CNNs) which are commonly used for image classification tasks, such as the CIFAR-10 dataset.

   TensorFlow provides a high-level API called Keras which simplifies the process of building neural networks. Keras allows developers to create a neural network model with just a few lines of code, making it an excellent tool for building and prototyping models. Additionally, TensorFlow provides a low-level API that enables users to have greater control over the implementation details of the model.

5.1.2 Implementation Details

The CIFAR-10 dataset in the project,is downloaded and preprocessed in a Google Colab using a deep learning framework such as TensorFlow. This preprocessing step included data augmentation techniques and improve the model's generalization performance. The preprocessed data is splited into training and validation sets and fed into the light-weight CNN model for training and evaluation.The model is trained at 100 epochs and track how our model is performing after each epoch of training.Finally, we see our model in action by visualizing some images from the test dataset and checked if our model predicts them correctly.

5.2 Testing

Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. Testing is a very important phase of system development. It is a phase in the software testing cycle where a total and integrated

application is tested. The scope of system testing is not only limited to the design of the system but also to the behavior and believed expectations of the business. There are different types of the testing. Thus, Testing is performed in order to meet the conditions, designing and executing of project and for checking results and reporting on the System process and performance.

5.2.1 Unit Testing

A unit test is a way of testing a unit - the smallest piece of code that can be logically isolated in a system. A unit testing is done during development of this system. In short Unit testing can be define as the testing process where each unit or modules are tested individually. The required inputs of the modules are given to it and the output is compared to the expected.

1. Test case 1: Upload image
2. Test Case 2: Detection of Objects

Test Case ID	Case1		
Test Case Description	Test case for uploading images		
Test Scenario	Successful upload of images in the system		
Test Data	uploading images		
Steps	Expected Results	Actual Result	Conclusion
upload of images in the system	User should be able to upload image in the file chosen section	Image Uploaded Successfully	Successfully

Table 5.1: Test Case for Uploading Images

Test Case ID	Case2		
Test Case Description	Test case for detecting images		
Test Scenario	Detection of the object based on the image provided.		
Test Data	Detecting image		
Steps	Expected Results	Actual Result	Conclusion
Object detection basedon the image provided	System should detect the object from the image provided and redirect to the result.	The class with the highest probability is selected as the predicted class for the input.	Predicted class is displayed to the user successfully.

Table 5.2: Test Case for detecting images

5.2.2 System Testing

System testing involves evaluating the performance of the web application in terms of uploading, storing, and displaying images.

Home Page for choosing file

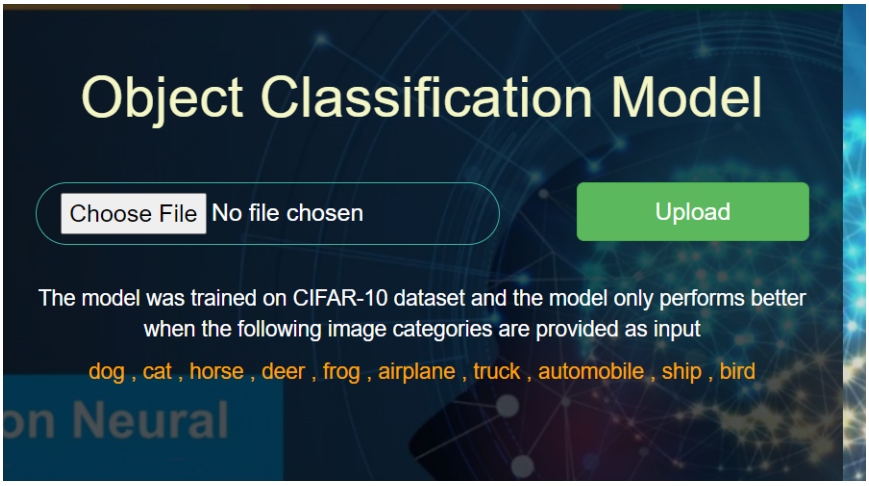


Figure 5.1: Home Page for Choosing File

Choosing File

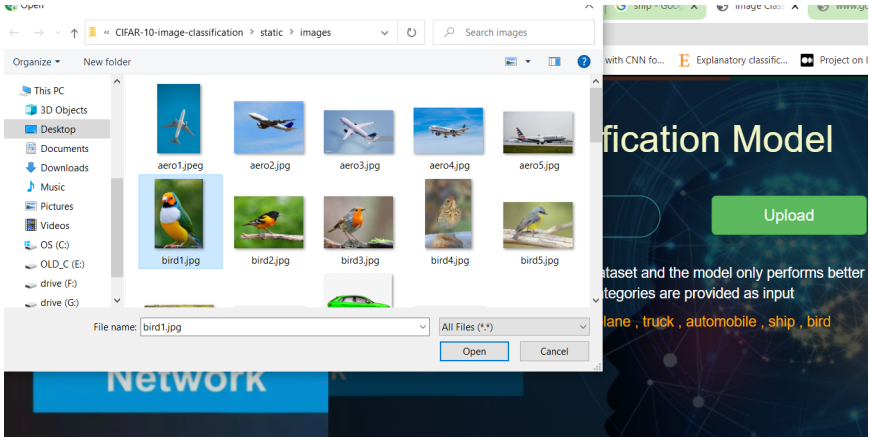


Figure 5.2: Choosing File

Files choosen

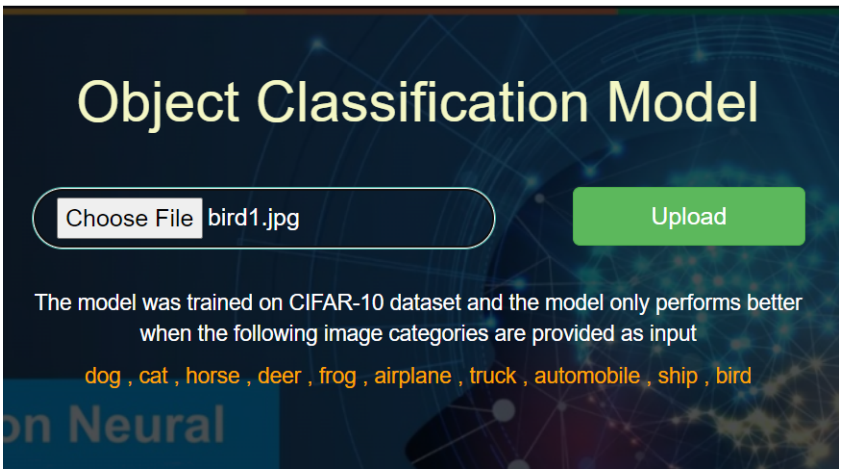


Figure 5.3: Files Choosen

Prediction

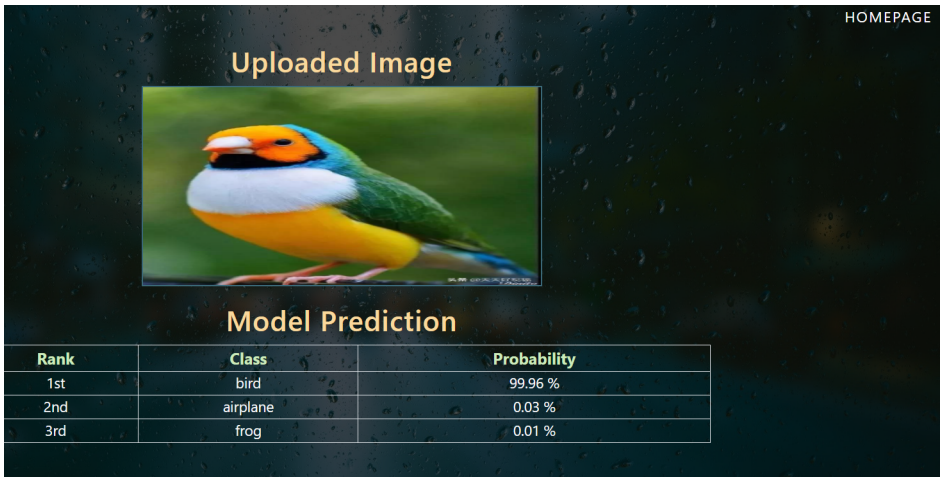
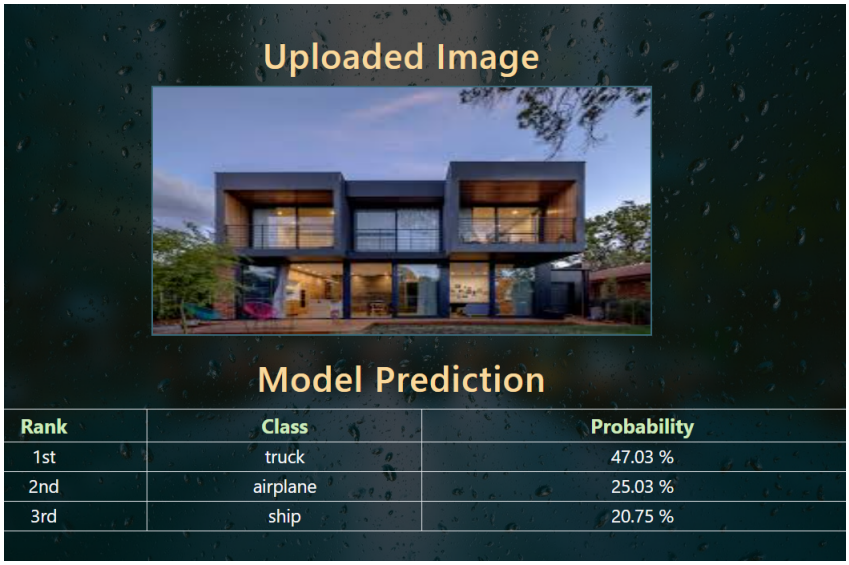
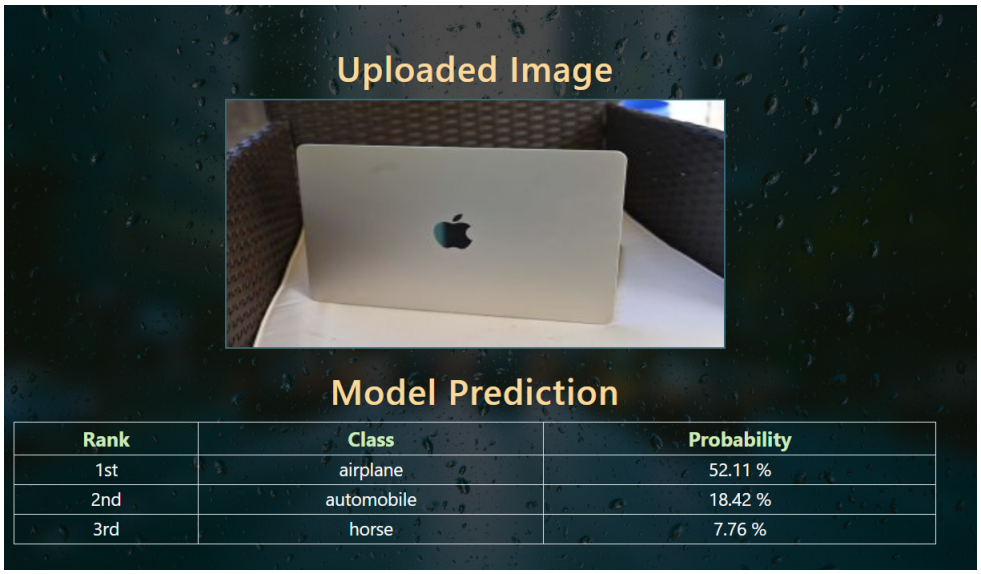


Figure 5.4: Prediction is Done

5.3 Limitations:

The project "Object Classification Using Lightweight CNN" has a limitation where if images of objects other than those present in CIFAR10 are inputted, the classification algorithm may output names that are similar to those in the CIFAR10 dataset but with lower probability.Here we uploaded normal images of a house, a laptop, and a tiger, respectively. However, these objects are beyond the scope of our predefined categories, so our model was unable to predict their true identity. Instead, the model classified them as objects that look similar to those belonging to the CIFAR10 dataset, resulting in incorrect predictions as shown in figures 5.5





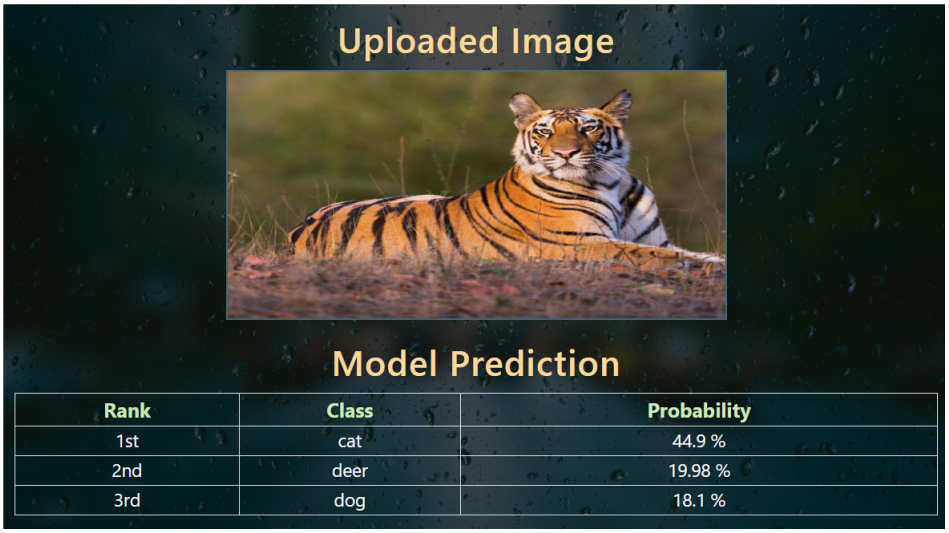


Figure 5.5: Prediction of other images

5.4 Result Analysis

We checkout performance analysis for various of Kernel.

Kernal size	Training Accuracy(%)	Training Loss(%)	Speed
1X1	0.77	0.64	29ms/step
3X3	0.98	0.05	43ms/step
5X5	0.97	0.09	40ms/step

Table 5.3: Result analysis for various of Kernel

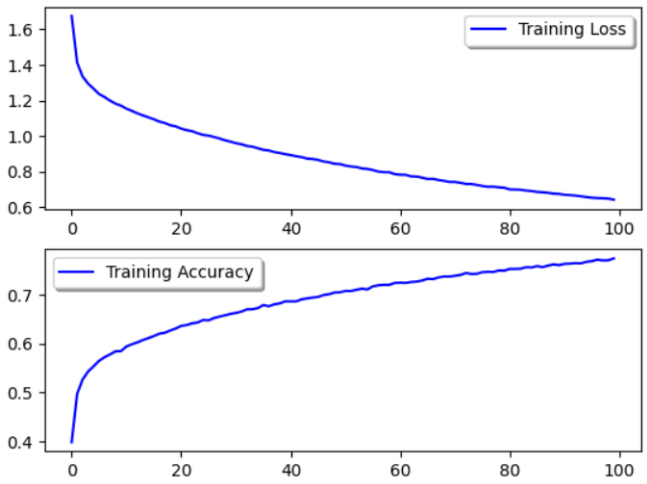


Figure 5.6: Training Accuracy and Loss of 1X1 Kernal

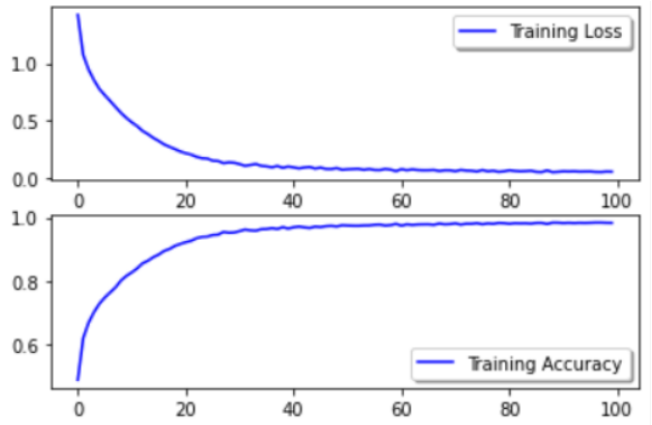


Figure 5.7: Training Accuracy and Loss of 3X3 Kernal

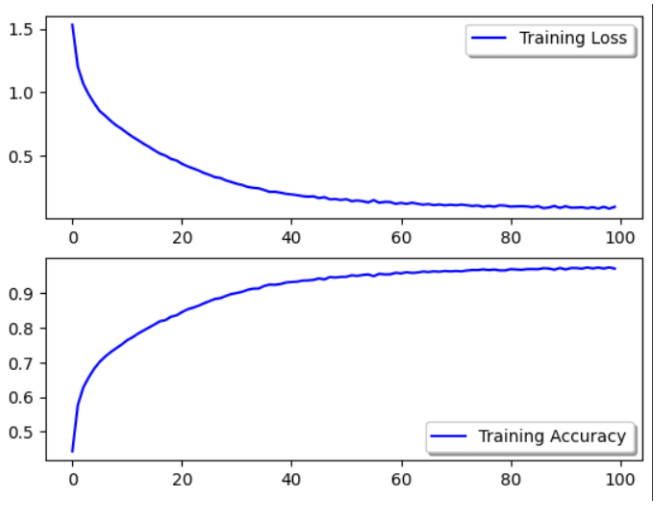


Figure 5.8: Training Accuracy and Loss of 5X5 kernal

We trained our model using kernel sizes of 1x1, 3x3, and 5x5, and found that the model achieved the highest accuracy with 98 percentage at 3x3 kernel size. Therefore, we selected this model as our final one.

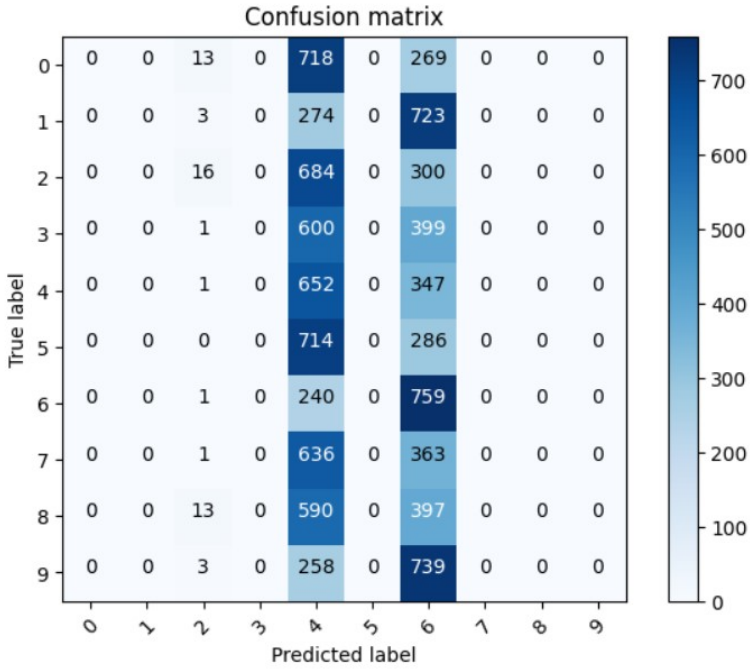


Figure 5.9: Confusion Matrix

Chapter 6 Future Recommendation and Conclusion

6.1 Future Recommendations

Here are some future recommendations for the project "Object Classification Using Lightweight CNN ":

1. **Use Data Augmentation:** One way to improve the model's accuracy is to use data augmentation techniques such as rotation, flipping, and zooming of images. This technique can help the model learn more robust features and reduce overfitting.

2. **Test on Larger Datasets:** While the Cifar10 dataset is useful for testing lightweight models, it would be interesting to test the model on larger datasets such as Cifar100, ImageNet, or COCO. This would provide a better understanding of the model's ability to generalize to different datasets and identify objects in more complex scenes.

6.2 Conclusion

The project "Object Classification Using Lightweight CNN " aimed to develop a deep learning model for classifying objects in images using the Cifar10 dataset. The project used a lightweight convolutional neural network (CNN) architecture and trained it on the Cifar10 dataset, which contains 60,000 32x32 color images in 10 classes.

After training and evaluating the model, the results showed that the architecture achieved an accuracy of 98 percent on the test set, which is a reasonably good result considering the simplicity of the network. The project also analyzed the training and validation accuracy and loss curves and concluded that the model did not suffer from overfitting.

Overall, the project successfully demonstrated the effectiveness of using lightweight CNN architectures for object classification tasks, particularly in situations where computational resources are limited. The project could be extended by testing the model on other datasets or by exploring other lightweight CNN architectures to improve accuracy further.

References

[1] G. Yao, T. Lei, and J. Zhong, "A review of convolutional-neural-network-based action recognition," *Pattern Recognition Letters*, vol. 118, pp. 14–22, 2019. [Online]. Available: https://doi.org/10.1016/j.patrec.2018.05.018

[2] M. B. Rozenwald, A. A. Galitsyna, G. V. Sapunov, E. E. Khrameeva, and M. S. Gelfand, "A machine learning framework for the prediction of chromatin folding in drosophila using epigenetic features," *PeerJ Computer Science*, vol. 6, p. e307, 2020. [Online]. Available: https://doi.org/10.1007/s13748-019-00203-0

[3] T. Kattenborn, J. Leitloff, F. Schiefer, and S. Hinz, "Review on convolutional neural networks (cnn) in vegetation remote sensing," *ISPRS journal of photogrammetry and remote sensing*, vol. 173, pp. 24–49, 2021.

[4] ——, "Review on convolutional neural networks (cnn) in vegetation remote sensing," *ISPRS journal of photogrammetry and remote sensing*, vol. 173, pp. 24–49, 2021.

[5] S. S. Basha, S. R. Dubey, V. Pulabaigari, and S. Mukherjee, *Neurocomputing*, 2020. [Online]. Available: https://doi.org/10.1016/j.neucom.2019.10.008

[6] R. Doon, T. Kumar Rawat, and S. Gautam, "Cifar-10 classification using deep convolutional neural network," pp. 1–5, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8745428

[7] R. Zaheer and H. Shaziya, "A study of the optimization algorithms in deep learning," in *2019 Third International Conference on Inventive Systems and Control (ICISC)*, 2019, pp. 536–539.

[8] A. Akwaboah, "Convolutional neural network for cifar-10 dataset image classification," 11 2019.

[9] [Online]. Available: https://www.kaggle.com/competitions/cifar-10

[10] J. Wu, "Introduction to convolutional neural networks," *National Key Lab for Novel Software Technology. Nanjing University. China*, vol. 5, no. 23, p. 495, 2017.

Annex I: Snapshots

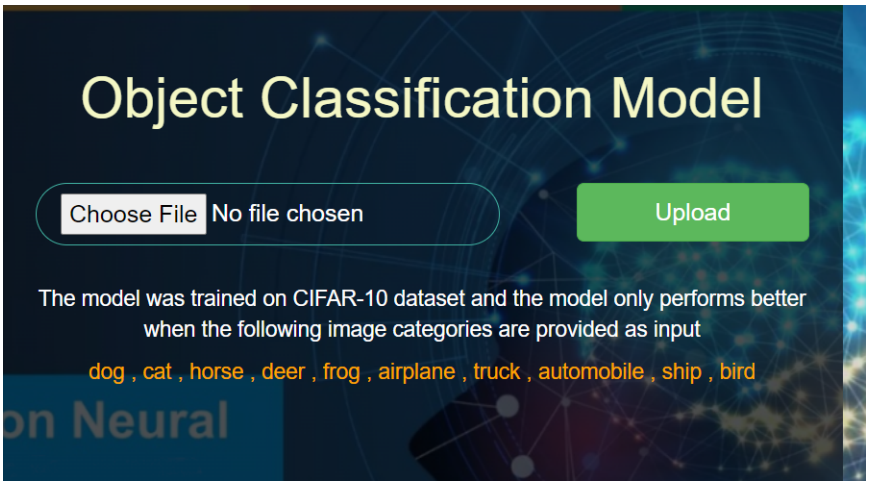


Figure 6.1: Homepage

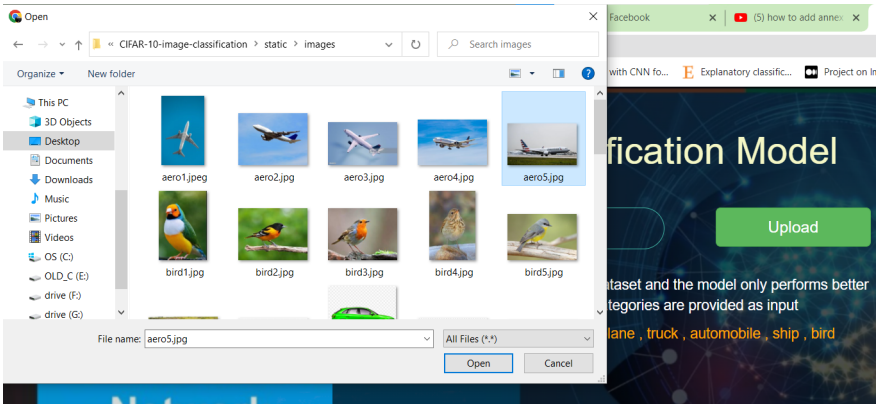


Figure 6.2: Image Upload Page

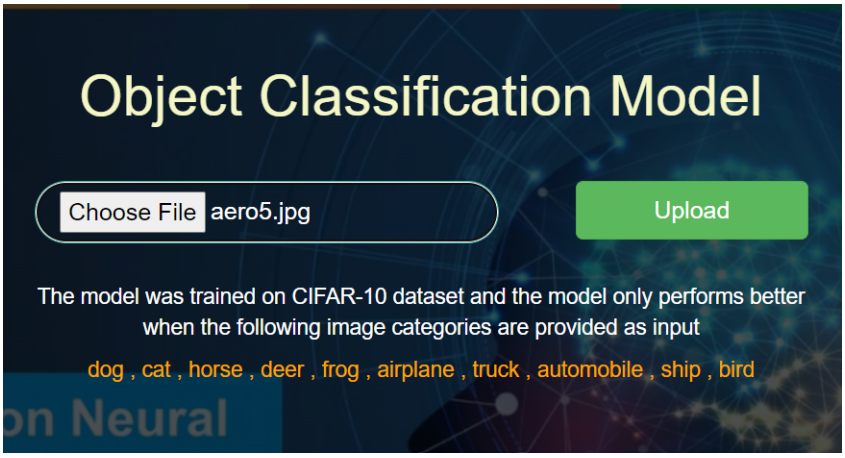
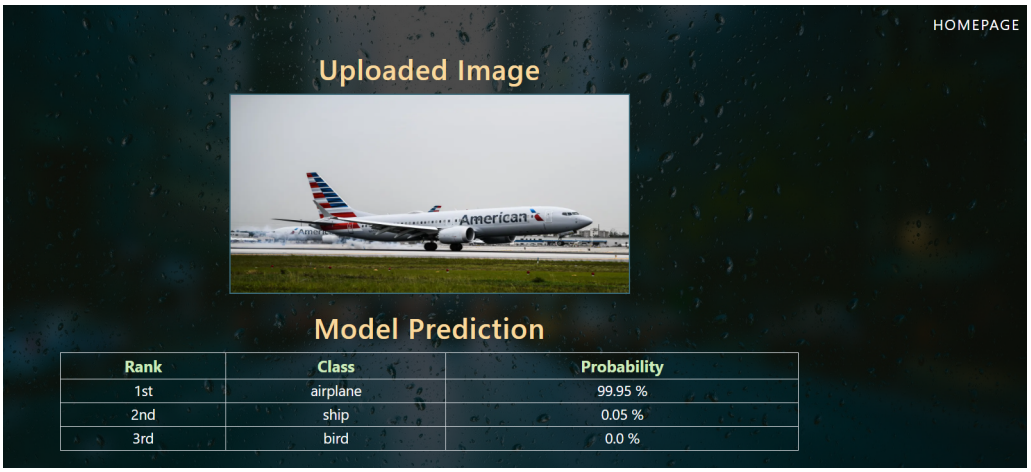
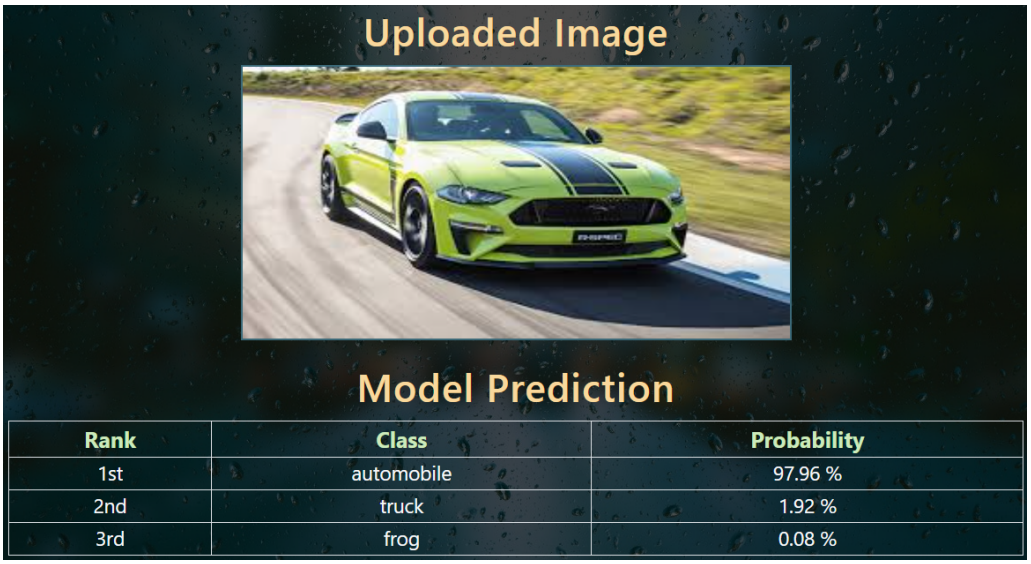


Figure 6.3: Image Uploaded Page





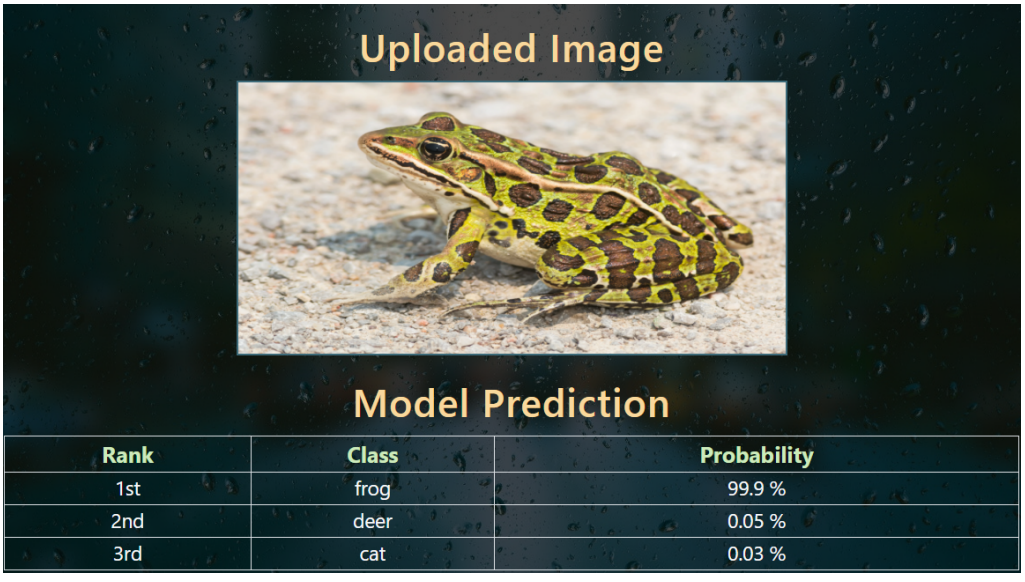
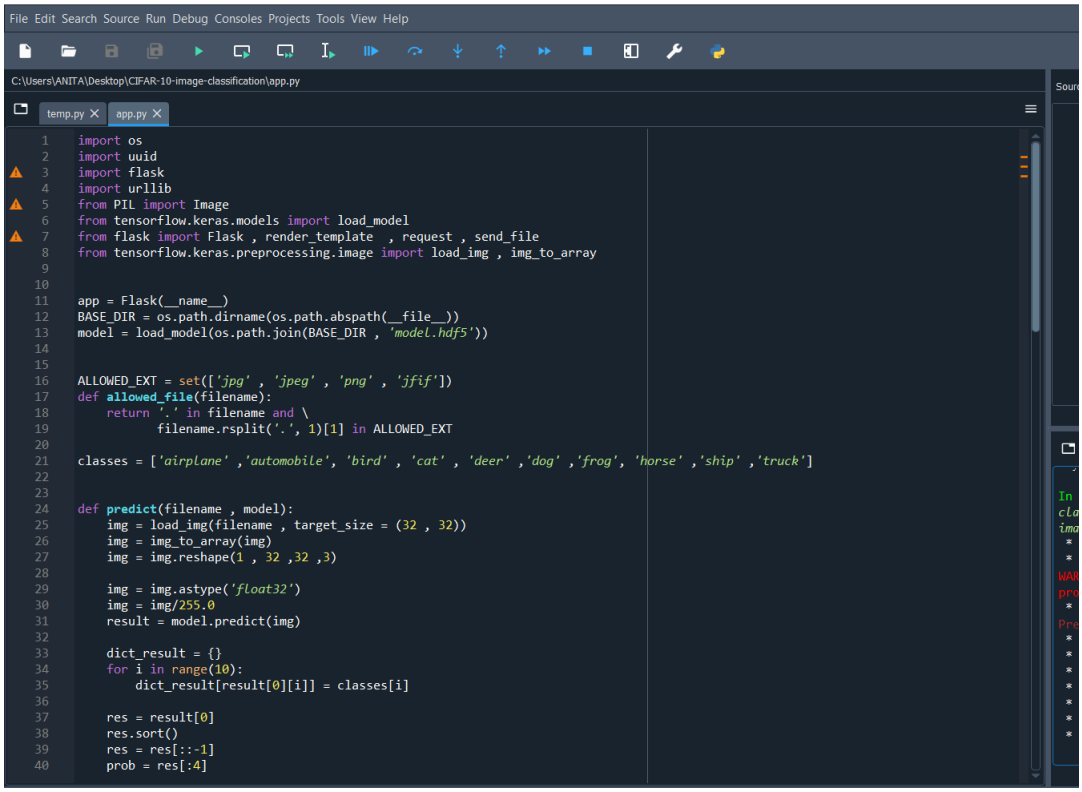


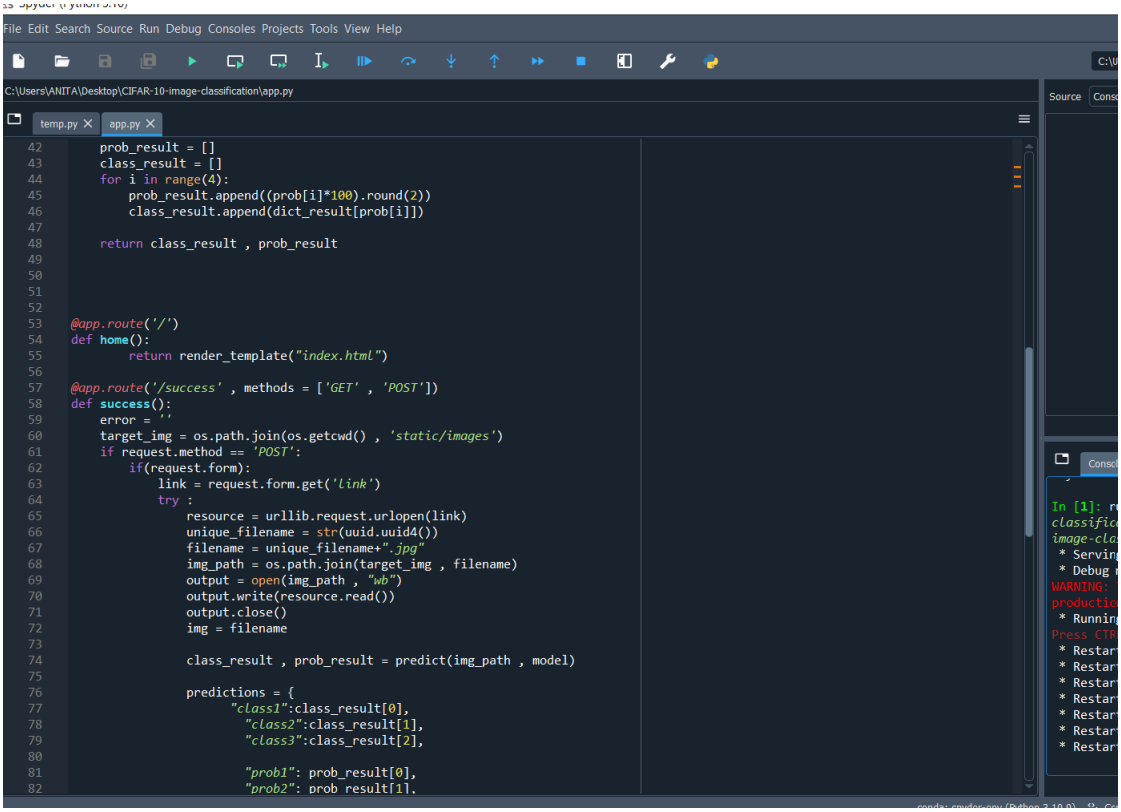
Figure 6.4: Prediction

Annex II: Snapshots

APP.PY Code



```
import os
import uuid
import flask
import urllib
from PIL import Image
from tensorflow.keras.models import load_model
from flask import Flask , render_template  , request , send_file
from tensorflow.keras.preprocessing.image import load_img , img_to_array


app = Flask(__name__)
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
model = load_model(os.path.join(BASE_DIR , 'model.hdf5'))


ALLOWED_EXT = set(['jpg' , 'jpeg' , 'png' , 'jfif'])
def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1] in ALLOWED_EXT

classes = ['airplane' ,'automobile', 'bird' , 'cat' , 'deer' ,'dog' ,'frog', 'horse' ,'ship' ,'truck']


def predict(filename , model):
    img = load_img(filename , target_size = (32 , 32))
    img = img_to_array(img)
    img = img.reshape(1 , 32 ,32 ,3)

    img = img.astype('float32')
    img = img/255.0
    result = model.predict(img)

    dict_result = {}
    for i in range(10):
        dict_result[result[0][i]] = classes[i]

    res = result[0]
    res.sort()
    res = res[::-1]
    prob = res[:4]
```



```
    prob_result = []
    class_result = []
    for i in range(4):
        prob_result.append((prob[i]*100).round(2))
        class_result.append(dict_result[prob[i]])

    return class_result , prob_result




@app.route('/')
def home():
        return render_template("index.html")

@app.route('/success' , methods = ['GET' , 'POST'])
def success():
    error = ''
    target_img = os.path.join(os.getcwd() , 'static/images')
    if request.method == 'POST':
        if(request.form):
            link = request.form.get('link')
            try :
                resource = urllib.request.urlopen(link)
                unique_filename = str(uuid.uuid4())
                filename = unique_filename+".jpg"
                img_path = os.path.join(target_img , filename)
                output = open(img_path , "wb")
                output.write(resource.read())
                output.close()
                img = filename

                class_result , prob_result = predict(img_path , model)

                predictions = {
                      "class1":class_result[0],
                        "class2":class_result[1],
                        "class3":class_result[2],

                        "prob1": prob_result[0],
                        "prob2": prob_result[1],
```

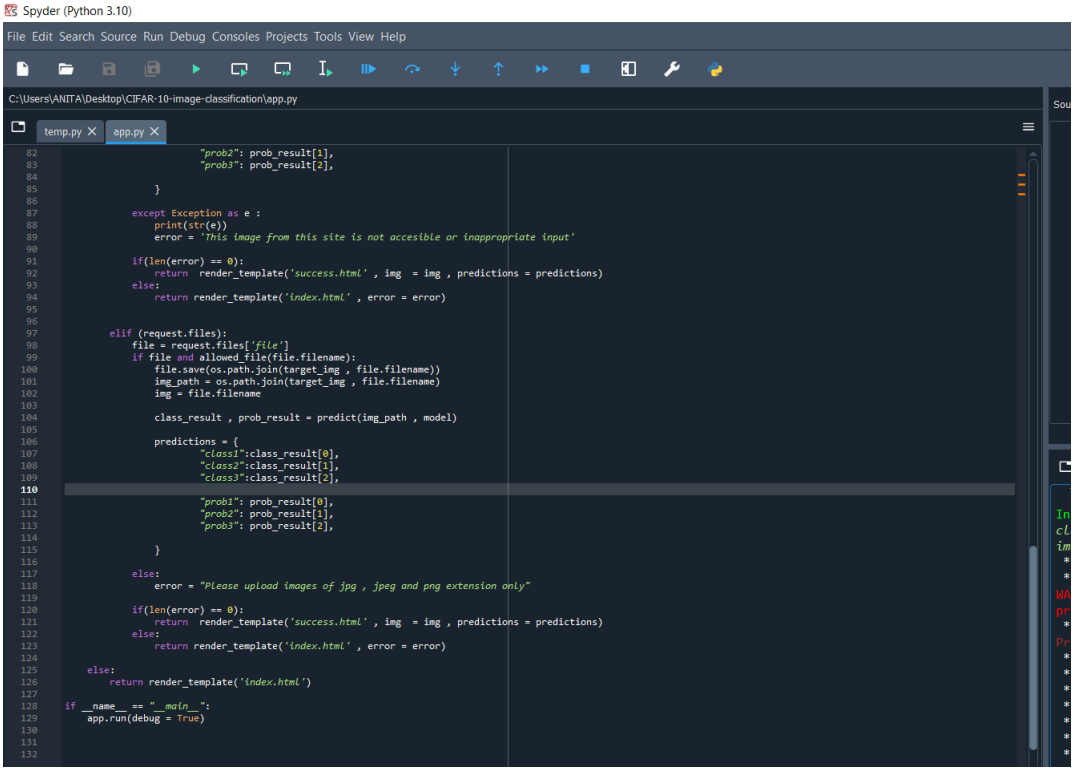
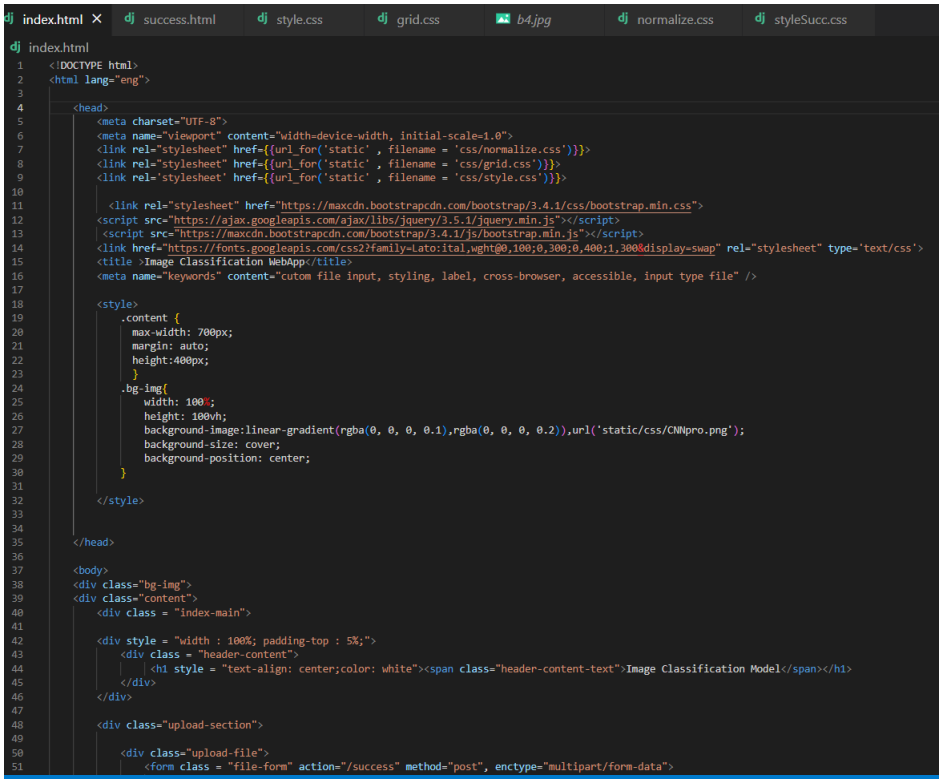


Figure 6.5: App.py

Frontend Source Code

index.html Code



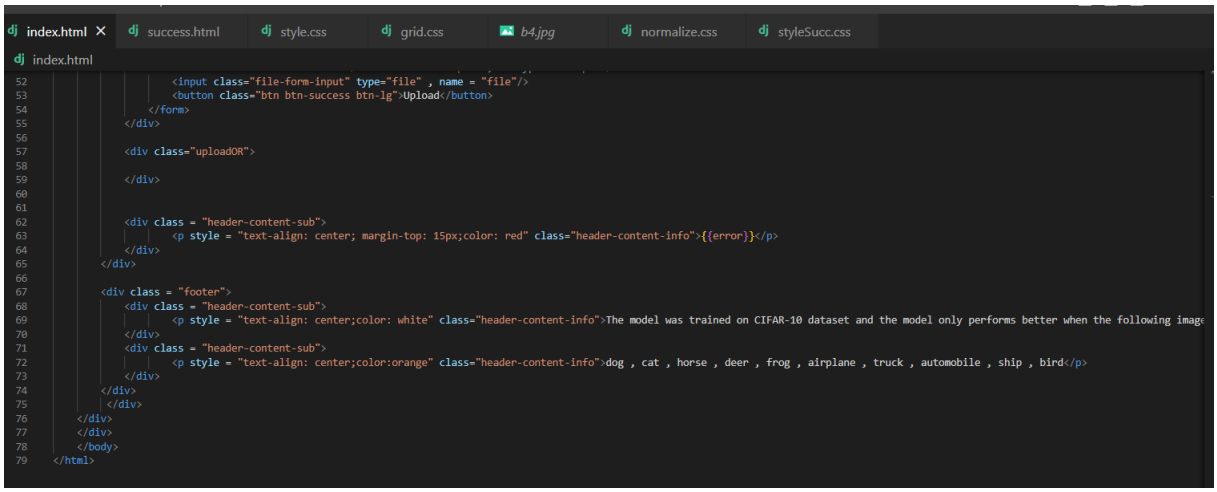
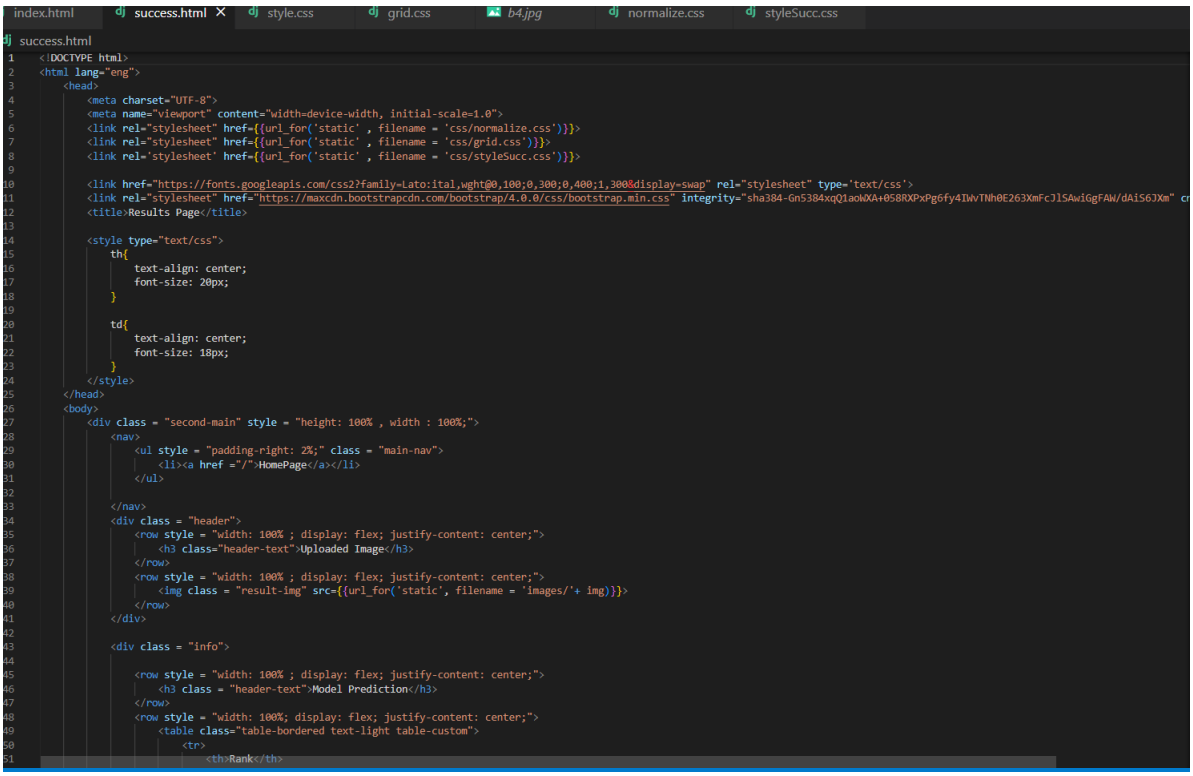
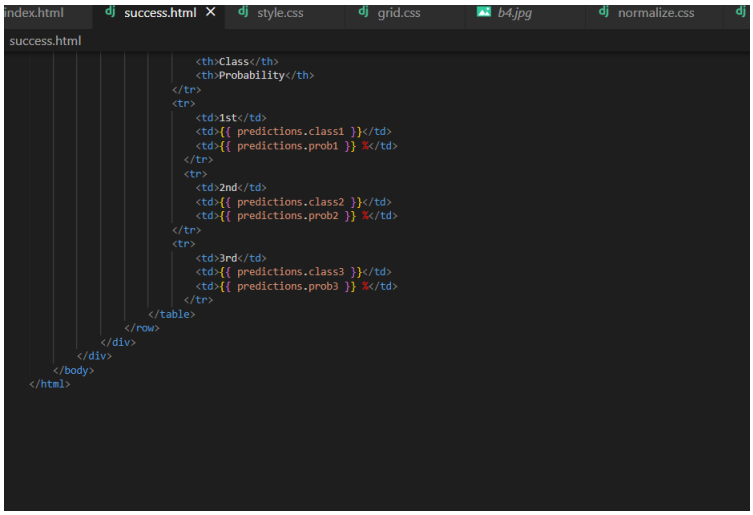


Figure 6.6: index.html

success.html Code



```
<!DOCTYPE html>
<html lang="eng">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" href={{url_for('static' , filename = 'css/normalize.css')}}>
        <link rel="stylesheet" href={{url_for('static' , filename = 'css/grid.css')}}>
        <link rel='stylesheet' href={{url_for('static' , filename = 'css/styleSucc.css')}}>

        <link href="https://fonts.googleapis.com/css2?family=Lato:ital,wght@0,100;0,300;0,400;1,300&display=swap" rel="stylesheet" type='text/css'>
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" cr
        <title>Results Page</title>

        <style type="text/css">
            th{
                text-align: center;
                font-size: 20px;
            }

            td{
                text-align: center;
                font-size: 18px;
            }
        </style>
    </head>
    <body>
        <div class = "second-main" style = "height: 100% , width : 100%;">
            <nav>
                <ul style = "padding-right: 2%;" class = "main-nav">
                    <li><a href ="/">HomePage</a></li>
                </ul>

            </nav>
            <div class = "header">
                <row style = "width: 100% ; display: flex; justify-content: center;">
                    <h3 class="header-text">Uploaded Image</h3>
                </row>
                <row style = "width: 100% ; display: flex; justify-content: center;">
                    <img class = "result-img" src={{url_for('static', filename = 'images/'+ img)}}>
                </row>
            </div>

            <div class = "info">

                <row style = "width: 100% ; display: flex; justify-content: center;">
                    <h3 class = "header-text">Model Prediction</h3>
                </row>
                <row style = "width: 100%; display: flex; justify-content: center;">
                    <table class="table-bordered text-light table-custom">
                        <tr>
                            <th>Rank</th>
```



```
                            <th>Class</th>
                            <th>Probability</th>
                        </tr>
                        <tr>
                            <td>1st</td>
                            <td>{{ predictions.class1 }}</td>
                            <td>{{ predictions.prob1 }} %</td>
                        </tr>
                        <tr>
                            <td>2nd</td>
                            <td>{{ predictions.class2 }}</td>
                            <td>{{ predictions.prob2 }} %</td>
                        </tr>
                        <tr>
                            <td>3rd</td>
                            <td>{{ predictions.class3 }}</td>
                            <td>{{ predictions.prob3 }} %</td>
                        </tr>
                    </table>
                </row>
            </div>
        </div>
    </body>
</html>
```

Figure 6.7: success.html

Data Train Code

projectCode.ipynb

File Edit View Insert Runtime Tools Help Last edited on April 13

Comment

+ Code + Text

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import tensorflow as tf
```

Load the **dataset**

```
[ ] (X_train, y_train),(X_test,y_test)=datasets.cifar10.load_data()
    X_train.shape
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 3s 0us/step
(50000, 32, 32, 3)
```

```
[ ] X_test.shape
```

```
(10000, 32, 32, 3)
```

Here we see there are 50000 training images and 1000 test images y_train is a 2D array, for our classification having 1D array is good enough.
so we will convert this to now 1D array.

```
[ ] y_train=y_train.reshape(-1,)
    y_train[:5]
```

```
array([6, 9, 9, 4, 1], dtype=uint8)
```

```
[ ] classes= ["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]
```

projectCode.ipynb

File Edit View Insert Runtime Tools Help Last edited on April 13

+ Code + Text

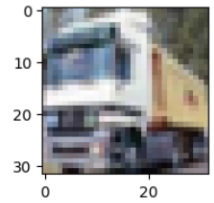
```
[ ] classes[9]
```

```
'truck'
```

Let's plot some images to see what they are:

```
[ ] def plot_sample(X,y,index):
      plt.figure(figsize=(15,2))
      plt.imshow(X[index])
      # plt.xlabel(classes[index])
```

```
plot_sample(X_train, y_train, 1)
```



Normalize the images to a number from 0 to 1. Image has 3 channels (R,G,B) and each value in the channel can range from 0 to 255. Hence to normalize in 0-->1 range, we need to divide it by 255

Normalizing the training data

projectCode.ipynb

File Edit View Insert Runtime Tools Help Last edited on April 13

+ Code + Text

Normalizing the training data

```
[ ] x_train=X_train/255
    x_test=X_test/255
```

Now let us build a convolutional neural network to train our images

```
cnn = models.Sequential([

    #cnn

    layers.Conv2D(filters=32, kernel_size=(1, 1), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(filters=64, kernel_size=(1, 1), activation='relu'),
    layers.MaxPooling2D((2, 2)),

     #dense layer
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

```
[ ] cnn.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        128

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 16, 16, 64)        2112

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 flatten (Flatten)           (None, 4096)              0

 dense (Dense)               (None, 64)                262208

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 265,098
Trainable params: 265,098
Non-trainable params: 0
```

projectCode.ipynb

File Edit View Insert Runtime Tools Help Last edited on April 13

+ Code + Text

```
[ ] cnn.compile(optimizer='adam',
                loss='sparse_categorical_crossentropy',
                metrics=['accuracy'])
```

```
[ ] classes = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

    plt.figure(figsize=(10,7))
    p = sns.countplot(y_train.flatten())
    p.set(xticklabels=classes)
```

\With CNN, at the end 100 epochs, accuracy was at around 98% which is a significant improvement over ANN. CNN's are best for image classification and gives superb accuracy. Also computation is much less compared to simple ANN as maxpooling reduces the image dimensions while still preserving the features

```
[ ] batch_size = 32
    num_classes = 10
    epochs = 100
```

```
history = cnn.fit(x_train, y_train, batch_size=batch_size,
                  epochs=epochs)
```

```
Epoch 72/100
1563/1563 [==============================] - 44s 28ms/step - loss: 0.7367 - accuracy: 0.7396
Epoch 73/100
1563/1563 [==============================] - 45s 29ms/step - loss: 0.7303 - accuracy: 0.7435
Epoch 74/100
1563/1563 [==============================] - 45s 29ms/step - loss: 0.7299 - accuracy: 0.7415
Epoch 75/100
1563/1563 [==============================] - 44s 28ms/step - loss: 0.7248 - accuracy: 0.7419
Epoch 76/100
1563/1563 [==============================] - 45s 29ms/step - loss: 0.7188 - accuracy: 0.7452
Epoch 77/100
1563/1563 [==============================] - 45s 29ms/step - loss: 0.7145 - accuracy: 0.7459
Epoch 78/100
1563/1563 [==============================] - 44s 28ms/step - loss: 0.7149 - accuracy: 0.7456
Epoch 79/100
1563/1563 [==============================] - 45s 29ms/step - loss: 0.7115 - accuracy: 0.7486
Epoch 80/100
1563/1563 [==============================] - 45s 29ms/step - loss: 0.7091 - accuracy: 0.7483
Epoch 81/100
1563/1563 [==============================] - 44s 28ms/step - loss: 0.6995 - accuracy: 0.7519
Epoch 82/100
1563/1563 [==============================] - 45s 29ms/step - loss: 0.6991 - accuracy: 0.7518
Epoch 83/100
1563/1563 [==============================] - 44s 28ms/step - loss: 0.6977 - accuracy: 0.7523
Epoch 84/100
1563/1563 [==============================] - 45s 29ms/step - loss: 0.6932 - accuracy: 0.7554
Epoch 85/100
```

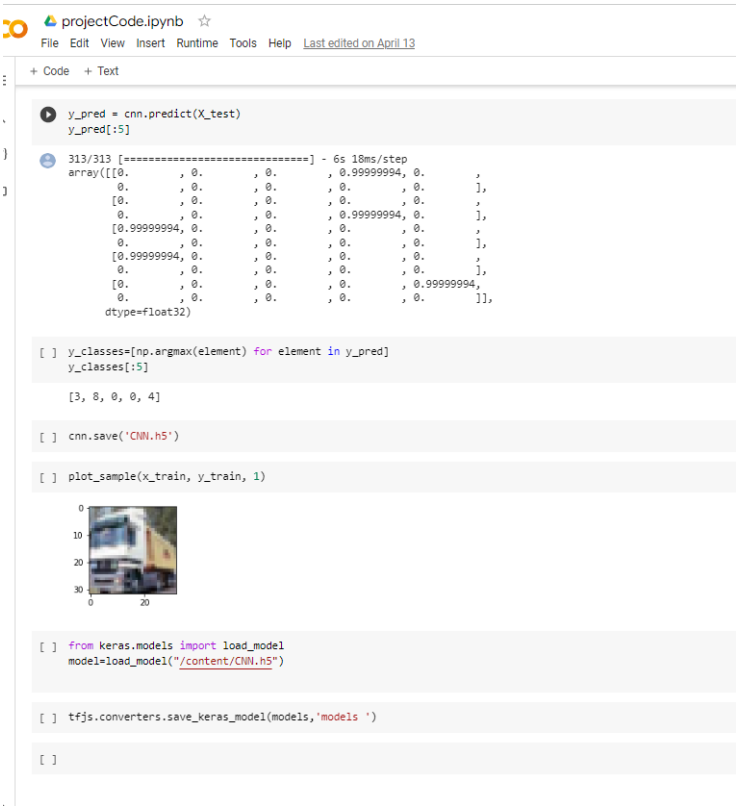


Figure 6.8: Data Train Code