



**TRIBHUVAN UNIVERSITY**

**INSTITUTE OF ENGINEERING**

**PASHCHIMANCHAL CAMPUS, POKHARA**

**Lamachaur, Pokhara-16**

**[Subject Code:755]**

**A MAJOR PROJECT REPORT ON**

**“MUSHROOM APP: IMAGE CLASSIFICATION INTO  
EDIBLE AND INEDIBLE”**

**Submitted by**

**Abishek Bhandari [PAS075BCT002]**

**Anup Adhikari [PAS075BCT006]**

**Ichchha Babu Bhattarai [PAS075BCT020]**

**Nitesh Pokhrel [PAS075BCT024]**

**Submitted to**

**Department of Electronics and Computer Engineering**

**Pashchimanchal Campus**

**May, 2023**

# **MUSHROOM APP: IMAGE CLASSIFICATION INTO EDIBLE AND INEDIBLE**

**Submitted by:**

**Abishek Bhandari      [PAS075BCT002]**

**Anup Adhikari      [PAS075BCT006]**

**Ichchha Babu Bhattarai [PAS075BCT020]**

**Nitesh Pokhrel      [PAS075BCT024]**

**Supervised by:**

**Urbara Bhandari**

**A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE  
REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER OR  
ELECTRONICS & COMMUNICATION ENGINEERING**

**Submitted to:**

**Department of Electronics and Computer Engineering**

**Pashchimanchal Campus**

**Lamachaur, Pokhara-16**

**May, 2023**

## **ACKNOWLEDGEMENT**

We would like to express our deepest appreciation and give thanks to each and every one who helped us during the completion of our project. We would also like to thank the Department of Electronics and Computer Engineering for providing a great environment and courses that helped us to gain knowledge about the various topics and providing an opportunity to work on this project.

This project would not have been possible without the continuous guidance and support of our supervisor Mrs. Urbara Bhandari, whose guidance and support helped us while we were stuck with nowhere to go. This project would not have been possible without his constant support and guidance. Her insightful comment helped us to improve our project.

## **ABSTRACT**

Worldwide, a large number of cases of harmful mushroom exposure and consumption results in hallucinations, sickness and even death. Due to morphological similarities between edible and inedible mushrooms, it is difficult for general public collectors to distinguish one from the other. So the main goal of our project is to develop a system that can classify user inputted mushroom images as edible or inedible. For this purpose the CNN model has been trained on 29,100 images of edible and inedible mushrooms where the ratio of training and testing data was 8:2. The model extracts influential features from the images implicitly and is able to distinguish between edible and inedible mushroom images. The model has been implemented in the mobile application built using flutter.

The smartphone application classifies user inputted pictures of mushrooms to edible or not edible, using a machine learning classifier based on a convolutional neural network (CNN) which is deployed in the server and output is fetched through api. It consists of 3 convolutional layers, each followed by a pooling layer, a fully connected layer and an output layer with a single node, outputting class index. Our app is a tool for deciding whether to collect a mushroom, not a tool for deciding whether to eat the mushroom. It helps people without proper mycology background or training to distinguish inedible mushrooms from edible ones with which they may be confused.

**Key Words:** Edible, Inedible, CNN, Smartphone application

# TABLE OF CONTENT

ACKNOWLEDGEMENT .....	iii
ABSTRACT.....	iv
CHAPTER 1: INTRODUCTION .....	1
1.1 BACKGROUND .....	1
1.2 PROBLEM STATEMENT .....	2
1.3 Objectives .....	3
CHAPTER 2: LITERATURE REVIEW .....	4
CHAPTER 3: METHODOLOGY .....	7
3.1 Software tools and technologies .....	7
3.1.1 Python .....	7
3.1.2 Tensorflow .....	7
3.1.2 Keras .....	7
3.1.3 Numpy.....	8
3.1.4 Pandas .....	8
3.1.5 Matplotlib.....	8
3.1.6 Flutter.....	8
3.1.7 Platform.....	9
3.2 Working Process .....	9
3.2.1 Image collection and dataset preparation.....	10
3.2.2 Data Pre-processing .....	11
3.2.3 Data Augmentation .....	12
3.2.4 CNN (Convolution Neural Network).....	12
3.2.5 CNN (Convolution Neural Network) Architecture.....	13
3.2.9 Training the network.....	22

CHAPTER 4: RESULT AND ANALYSIS.....	24
4.1 Experiment.....	24
4.2 Result analysis .....	24
4.2.1 Evaluation Metrics .....	26
4.3 Model predictions .....	28
CHAPTER 5: CONCLUSION .....	29
REFERENCES .....	30

## LIST OF FIGURES

<b>Figure 3.1 :</b> Workflow diagram .....	9
<b>Figure 3.2 :</b> Dataset Distribution .....	10
<b>Figure 3.3 :</b> Mushroom Dataset .....	11
<b>Figure 3.4 :</b> Data Augmentation .....	12
<b>Figure 3.5 :</b> Convolution Neural Network .....	13
<b>Figure 3.6 :</b> CNN Architecture .....	1
<b>Figure 3.7 :</b> A 6x6x1 image convolved with a 3x3x1 kernel to become a 4x4x1 convolved feature. ....	17
<b>Figure 3.8 :</b> Representation of Max Pooling layer .....	19
<b>Figure 3.9 :</b> Sigmoid Function .....	20
<b>Figure 3.10 :</b> Re- LU Activation .....	21
<b>Figure 3.11 :</b> Screenshoot of mobile UI .....	23
<b>Figure 4.1:</b> Training and testing accuracy plot for 10 epoches .....	24
<b>Figure 4.2:</b> Training and testing accuracy plot for 50 epoches .....	25
<b>Figure 4.3:</b> Training loss vs epochs .....	25
<b>Figure 4.4 :</b> Confusion matrix .....	26
<b>Figure 4.5 :</b> Evaluation Metrics .....	26
<b>Figure 4.6 :</b> Model Prediction .....	28

## LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
RGB	Red Green Blue
ReLU	Rectified Linear Unit
ELU	Exponential Linear Unit
RAM	Random Access Memory
CPU	Central Processing Unit
GB	GigaByte
NN	Neural network
SVM	Support Vector Machine
KNN	K-Nearest Neighbour
CLAHE	Contrast Limited Adaptive Histogram Equalization
AHE	Adaptive Histogram Equalization
Adam	Adaptive Moment Estimation
DL	Deep learning



# CHAPTER 1: INTRODUCTION

## 1.1 BACKGROUND

Mushroom is the fleshy fruit body of different classes of fungus which is adherent to Basidiomycetes growing on a ground surface. Nowadays, mushrooms are popular valuable foods because they are low in calories, rich in fiber, minerals and nutrients with copper, magnesium, potassium, zinc; also, they are cholesterol-free. But all mushrooms are not safe to consume. There are almost 14000 species of mushrooms present in the world out of which only some are edible. However, morphological similarities between edible and poisonous species often result in the unintended consumption of harmful fungi with severe or even fatal consequences. For instance, Amanita is a genus of poisonous mushroom species. According to reports, a person consuming only five fried slices of Amanita can experience symptoms including central nervous system depression, ataxia, waxing, waning obtundation, religious hallucinations, and hyperkinetic behavior. Other mushrooms can also be highly toxic. Mushroom poisoning is not new to Nepal. Every year dozens of people are admitted to hospital for treatment, while hundreds more rely on local treatments due to mushroom poisoning. During the rainy season, many people rely on wild mushrooms as a much-needed food source and also as a flavorful addition to their diet. Whole families have been wiped out by consuming poisonous wild mushrooms. Many are not afraid of using wild mushrooms despite knowledge of the risks associated with the poisonous effects of some mushrooms.

Humans are expert at classifying objects based on attributes and have been recognizing poisonous mushrooms considering the shape, cap diameter, stem diameter, color, odor, location and secretion skins by experience for many years. Inedible mushrooms have irregular shaped cap and stem, are brightly coloured, have foul smells while edible mushrooms have smooth and dry caps, well defined stem, come in colors like white, brown, have pleasant earthy aroma. But the accuracy of this intuitive method is very low and the annual poisoning events have proven it. Thus, it's not a solid technique for recognizing whether mushrooms are toxic or not.

So, our smartphone application aims to classify user inputted images of mushrooms to edible or not edible, using a machine learning classifier based on a convolutional neural network (CNN). CNNs are the current state-of-the-art in image recognition that take an image as input, assign weights and biases to all the aspects of an image and thus differentiates one from

the other. The network is trained by using batches of images, each of them having a label to identify the real nature of the image. A batch can contain a few tenths to hundreds of images. For each and every image, the network prediction is compared with the corresponding existing label, and the distance between network prediction and the truth is evaluated for the whole batch. Then, the network parameters are modified to minimize the distance and thus the prediction capability of the network is increased. The training process continues for every batch similarly.

To train such a classifier, large amounts of pictures of previously classified mushrooms were used as examples. We cannot simply use any pictures of mushrooms, since they have to include a reliable edibility tag. The size and shape of pictures were normalized, since usually classification algorithms expect pre-specified input sizes. We only used images as input, so features such as smell and location were not be used by our classifier. The CNN Model performs feature extraction implicitly based on an input image, using its convolutional layers. It is simply a matter of defining the structure of the network in such a way that the network is able to extract the information it needs for the classification task. Instead of simply outputting a binary class, the model outputs the probability of inedibility.

## **1.2 PROBLEM STATEMENT**

Nepal is rich in mushroom diversity. Local people have been using wild mushrooms in their diet as well as a source of income, but they do not have proper knowledge about identification of edible and poisonous mushrooms. This practice has caused severe poisonings and even death. For many, the negative effects of wild mushrooms outweigh their positive value. The reports of death from the various parts of Nepal come at an alarming rate, but still a large proportion of such incidents go unreported by the local news. On the other hand, due to the availability of different types of mushrooms in Nepal and the similar structure of some mushrooms, it is very difficult to distinguish them with the naked eye. If we consume poisonous or inedible mushrooms instead of edible mushrooms then it can cause harmful effects in our body. For instance, in a reported case, a 53-year-old consumed *Gyromitra* and experienced vomiting and diarrhoea, subsequently expressing hypotension, anuria, jaundice, hemiplegia, and coma, followed by death on the 3rd day. For this reason, correctly identifying mushrooms in their own classes at the appropriate time is also essential to preventing negative effects from the consumption of poisonous mushrooms.

### **1.3 Objectives**

The main objective of our project is:

1. To develop a smartphone application to classify mushroom images to edible/inedible.

## CHAPTER 2: LITERATURE REVIEW

Jae Joong lee and others in [1] have proposed a method to classify mushroom types from field collection images using a smartphone application based on a convolutional neural network. An android app was developed by transferring the server-based trained model allowing users to obtain probability scores for the correct genus classification. Three models were developed to classify easily confused genera. First model classified species of the morphologically similar genera of *Gyromitra* (poisonous species) and *Morchella* (highly sought edible species). The second model was able to classify genera of *Clavulina*, *Inocybe*, and *Marasmius*, and the third model was able to classify genera of *Agaricus* (some edible species), *Amanita* (includes deadly species), *Cantharellus* (prized edibles), *Pleurotus* (prized edibles), and *Tricholoma* (some toxic species). The first two models relied on the same neural network architecture, but the third model employed a different one. A convolutional neural network and a feed-forward neural network comprised the architecture of the first two models. For the convolutional neural network, input variables were  $224 \times 224$ -pixel images in three-channel RGB color and six convolutional layer blocks. Max-pooling was applied after 1st, 3rd, 4th, and 5th convolutional layers. A rectified linear unit (ReLU) activation function was used after the six convolutional layers. The feed-forward network consisted of three fully connected layers; the number of features in the last layer was equivalent to the number of classes. After each fully connected layer, exponential linear unit (ELU) activation was used with 70% dropout rates. This neural network architecture was unable to obtain satisfactory classification results for the 5-class model. Therefore, the authors decided to utilize the fine-tuned ResNet system with 152 layers model and added an output layer with a number of neurons equivalent to the number of mushroom species classes. The hardware specification of the server was Intel Xeon CPU @ 2.30GHz, 34GB RAM, and Tesla P100-PCIE-16GB.

In the paper [2] Mohammad Ottom and others have classified mushrooms into two categories poisonous and non-poisonous using machine learning algorithms like neural network (NN), SVM, Decision Tree, and KNN on different scenarios, with background and without background. They used Orange3 & Knime to build a machine learning model. They extracted different features from mushroom images like Eigen features, histogram features and parametric features using MATLAB. In order to improve the results, they removed the background but unfortunately this step failed to improve the result. Finally, the experiment

results showed advantage for background images, especially when used KNN algorithm, and with Eigen features extraction and real dimensions of mushroom (i.e cup diameter, stem tall and stem diameter) accuracy reached to 0.944, while the result after replacing real dimensions with virtual dimension (i.e., width and height of mushroom shape inside the images) is 87%. The highest value for KNN after removing images background reached to 0.819 as a maximum value.

In the paper [3] Nusrat Zahan has applied InceptionV3, VGG16, and Resnet50 on 8190 mushroom images where the ratio of training and testing data was 8:2. The collected images were not suitable in size and color and some were noisy. A contrast enhanced method CLAHE was applied to improve the image quality. CLAHE is a variant of Adaptive Histogram Equalization (AHE) which improves the local contrast-enhancing of images and the definitions of edge. A large number of the dataset was needed to train the model but the collected data was insufficient. For this reason, the overfitting problem arises and to overcome it, he has used data augmentation techniques like rotation, translation, scaling and shearing. The Contrast Limited Adaptive Histogram Equalization (CLAHE) method was used along with InceptionV3 to obtain the highest test accuracy. Vgg16, Resnet50, and InceptionV3 are some of the prevailing CNN architectures used in this experiment. Among all these CNN architectures, VGG16 came out with the best accuracy of 84.2% while using raw images where InceptionV3 and Resnet50 achieved 82.31 and 53.25 respectively. In contrast, InceptionV3 outperforms any remaining structures with a different upgraded dataset using contrast enhancement technique with an accuracy of 88.40% followed by VGG16 with a precision.

Teemu Koivisto and Tuomo Niemien have described a smartphone application which classifies user inputted pictures of wild mushrooms to edible or not edible, using a machine learning classifier based on a convolutional neural network (CNN). They only used images as input, so features such as smell and location were not used by the classifier. The mushrooms images were retrieved from the mushroom world website using web scrapers. They were only able to retrieve 536 images so they artificially increased the number of images by augmentation. The CNN model described in the paper consists of 3 convolutional layers, each followed by a pooling layer, a fully connected layer and an output layer with a single node, outputting a probability value. Instead of simply outputting a binary class, the model outputs the probability of edibility. The image data was split to training and testing sets with a split of 80% to 20%.

The CNN was then trained by feeding it small batches of augmented image data from the training set, 32 images at a time. Each slightly altered image was passed through the network 100 times, i.e. the network was trained on 100 epochs. The training accuracy has an increasing trend, the testing accuracy averages a steady 55% accuracy, which is remarkably poor.

In the paper “Classification of Mushrooms to Detect their Edibility Based on Key Attributes” authors have used a dataset containing 8124 instances and 22 attributes of mushroom obtained from UCI Machine Learning Repository for classifying the mushroom into edible and poisonous. Then the dataset is preprocessed and it is analyzed using Weka and a decision tree is also constructed for the dataset using data mining techniques. The data mining tool, WEKA (Waikato Environment for Knowledge Analysis), includes a collection of data mining algorithms and also contains options for data preprocessing, clustering, classification, regression, visualization. [4]

Chowdhury and S. Ojha in [5] identified a manner to distinguish several mushroom diseases using different data mining classification methods. They used actual dataset gathered from mushroom farms by using data mining like Naive Bayes, RIDOR and SMO algorithms. The Mushroom dataset consists of 110 instances and 19 attributes including classification attributes. They performed comparison based on a statistical way to detect popular symptoms for mushroom to discover mushroom disease. They concluded that naïve Bayes gives best results with comparisons to other classification techniques.

## **CHAPTER 3: METHODOLOGY**

### **3.1 Software tools and technologies**

For the completion of the project we have used following programming languages,libraries, frameworks for the development of our system.

#### **3.1.1 Python**

Python is a high-level, interpreted programming language that is easy to learn and widely used in various domains of software development. It was created by Guido van Rossum and first released in 1991.It provides a fantastic ecosystem for machine learning and data science. Python provides a significant number of libraries that enhance development skills and provide an expert community to tackle specific difficulties.

#### **3.1.2 Tensorflow**

TensorFlow is an open-source software library that is widely used for machine learning and artificial intelligence applications. It allows for the creation and training of neural networks, deep learning models, and natural language processing models using a data flow graph. TensorFlow offers a wide range of pre-built operations and functions that simplify the process of model building, as well as tools for debugging and visualization. Its flexibility and portability enable it to be used across various platforms, including CPUs, GPUs, and mobile devices. TensorFlow is supported by a large and active community of developers and researchers, which provides resources, tutorials, and models that can be used by anyone

#### **3.1.2 Keras**

Keras is an open-source software library written in Python that provides a high-level interface for building and training neural networks.It is constructed on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML as a backend and is intended to be user-friendly, flexible, and extendable.With just a few lines of code, Keras enables programmers to create sophisticated deep learning models. Convolutional neural networks, recurrent neural networks, and deep neural networks may all be easily created using its user-friendly and intuitive API for defining models.Additionally, Keras has built-in support for typical deep learning operations including activation functions, loss functions, and optimizers.

### **3.1.3 Numpy**

NumPy is a powerful Python library for numerical computing that is widely used in scientific computing, data analysis, and machine learning. It provides an efficient and convenient way to work with arrays and matrices of numerical data, which makes it a popular choice for numerical operations such as matrix multiplications, mathematical operations, and statistical analysis. The best thing about numpy is its performance .

### **3.1.4 Pandas**

Pandas is an open-source Python library used for data manipulation and analysis. It provides a powerful and flexible toolset for working with structured data, including data cleaning, preparation, exploration, and visualization. The DataFrame, a two-dimensional table-like structure that can hold both numerical and categorical data, is the primary data structure used by Pandas. It offers a variety of tools and operations for manipulating data, including filtering, joining, and grouping data as well as merging, joining, and altering it. Pandas handles the missing values . It offers resources for locating and dealing with missing values as well as tools for impute missing data using various strategies, like interpolation. It easily interfaces with other Python libraries like NumPy and Matplotlib. It can also handle a variety of data sources, including CSV, Excel and SQL databases

### **3.1.5 Matplotlib**

Matplotlib is a NumPy-based multi-platform data visualization framework presented by John Hunter in 2002. It provides a wide range of tools for creating high-quality plots, charts, and graphs for data analysis and presentation. Matplotlib is highly customizable, allowing users to create plots that fit their specific needs.

### **3.1.6 Flutter**

Flutter was developed and is currently maintained by Google. The initial version of Flutter was unveiled at the Dart developer summit in 2015, and it was officially launched as a stable release in December 2018. Using a single codebase, developers can create high-performance, natively built applications for mobile, web, and desktop using the open-source UI toolkit Flutter. The reactive programming paradigm, which enables quick and flexible UI development, the customisable widget library, which offers a variety of pre-built UI components, and hot reload



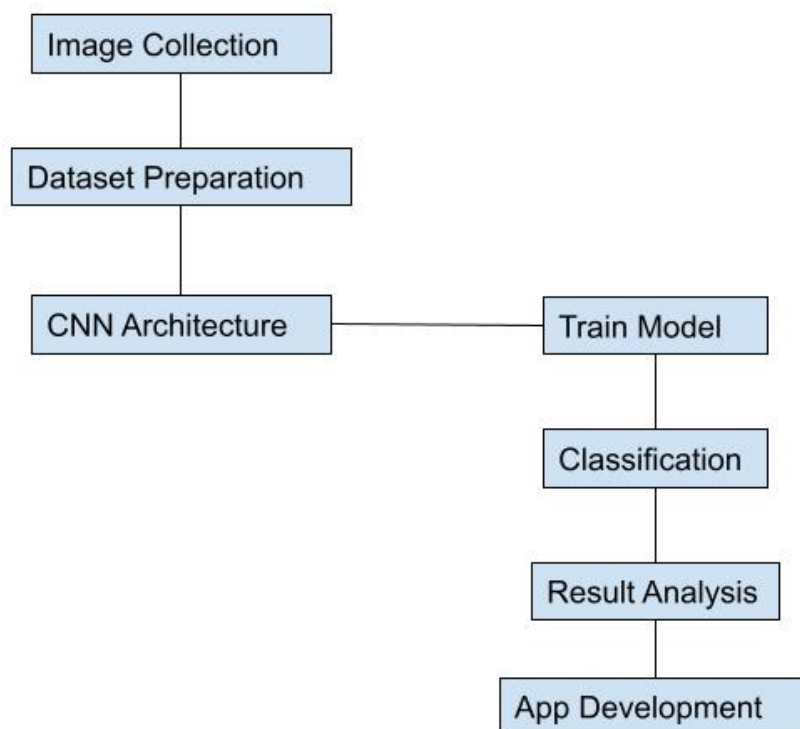
are some of its important characteristics. The latter is an effective tool that enables developers to view changes made to the code in real-time. Flutter is a well-liked option for developers wishing to build stunning and responsive apps across a variety of devices since it also provides excellent documentation, a growing ecosystem of plugins and packages, and solid support for testing and debugging.

### 3.1.7 Platform

To manage our project code we use android studio for the development of flutter app. For the training and testing of machine learning models, a jupyter notebook is used in the local machine. But Google Collab is preferred over Jupyter notebook. Colab provides a generous amount of resources, including CPU, RAM, and GPU, depending on availability. We used the GPU to train our model .

## 3.2 Working Process

Overall process consists of the five main activities from data collection to the analysis of the result that the model predicts.

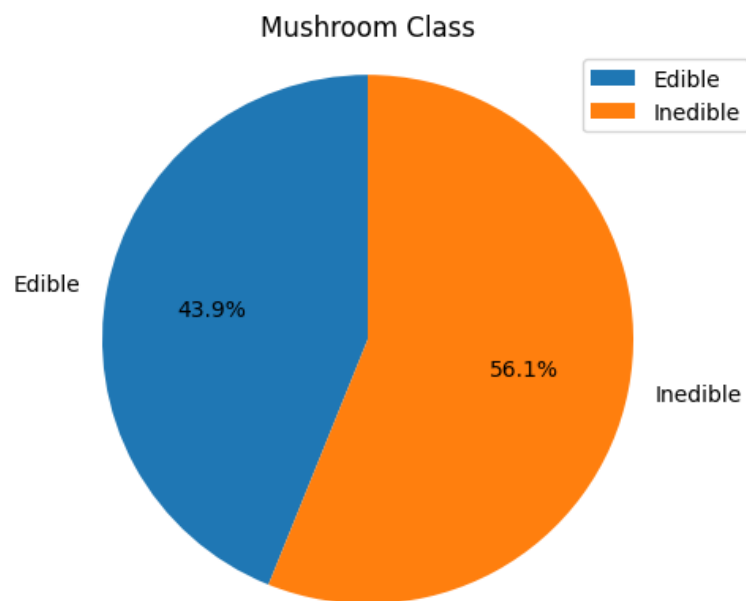


**Figure 3.1 :** Workflow diagram

### 3.2.1 Image collection and dataset preparation

We have used MO\_106 dataset consisting of 29100 images categorized into 106 mushroom species. The dataset contains a csv file describing the edibility status of each species. Based on the tag edible species image are grouped together to create edible image set while inedible species are grouped together to create inedible image set. In this way we created a binary classification task. Finally the images were splitted into training and testing sets with 8:2 split ratio. The training set consists of 10,220 edible images and 13,060 inedible images while testing set consists of 2555 edible images and 3265 inedible images.

The visualization is shown below in the form of a pie-chart.



**Figure 3.2 : Dataset Distribution**



**Figure 3.3 : Mushroom Dataset**

### 3.2.2 Data Pre-processing

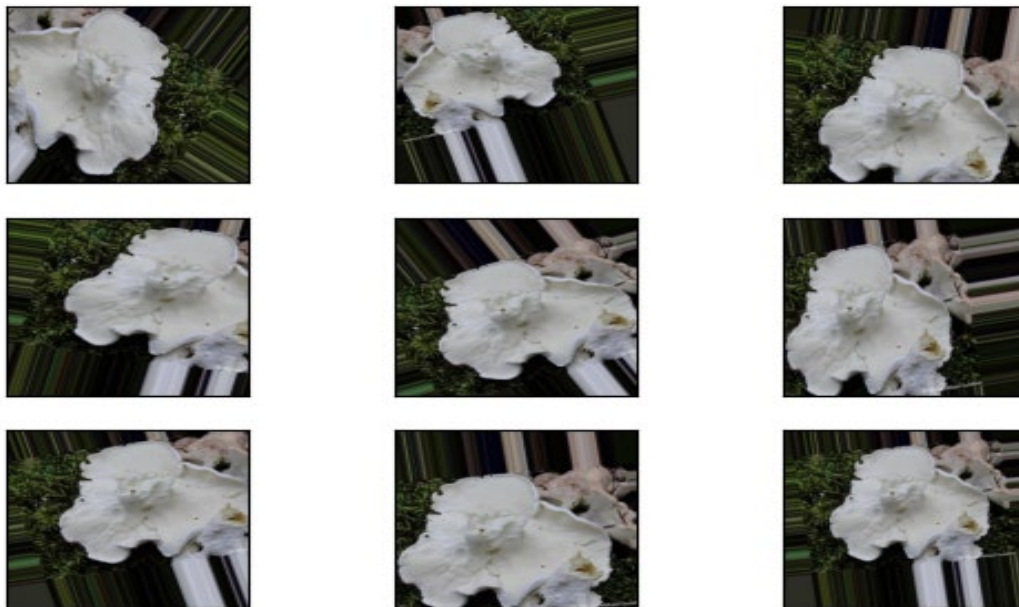
To ensure uniformity in the dataset, we opted to standardize the images, which initially exhibited significant variations in size and quality. Given their original high-quality nature, the images occupied considerable storage space. In order to address this, we resized the images to a consistent dimension of 64\*64 pixels

This process allowed us to maintain the original aspect ratios while ensuring all images shared the same dimensions. The images in the dataset are in color, consisting of three stacked matrices (each measuring 64\*64 pixels) representing the red, green, and blue color channels. The pixel values within these matrices originally ranged from 0 to 255. However, to prevent potential numerical issues during model training, we applied a normalization technique. This involved rescaling the pixel values to a new scale ranging from 0 to 1. By performing this normalization, we ensured that the pixel values were represented in a standardized range suitable for training the model effectively.

Overall, these preprocessing steps allowed us to achieve consistent image sizes, reduce storage requirements through compression, and normalize the pixel values for improved model training outcomes.

### 3.2.3 Data Augmentation

When training deep learning models, having a large amount of diverse data is generally beneficial for achieving good performance. However, if the dataset is relatively small, it can limit the model's ability to learn and generalize patterns effectively. In such cases, data augmentation techniques are used to artificially increase the number of unique training examples. In the context of our mushroom dataset, we used the Keras deep learning library to create an image generator. Instead of using the original images directly, this generator produces randomly altered images. These alterations can include various transformations such as rotations, translations, scaling, flips. Data augmentation techniques help to reduce model overfitting and improve accuracy.

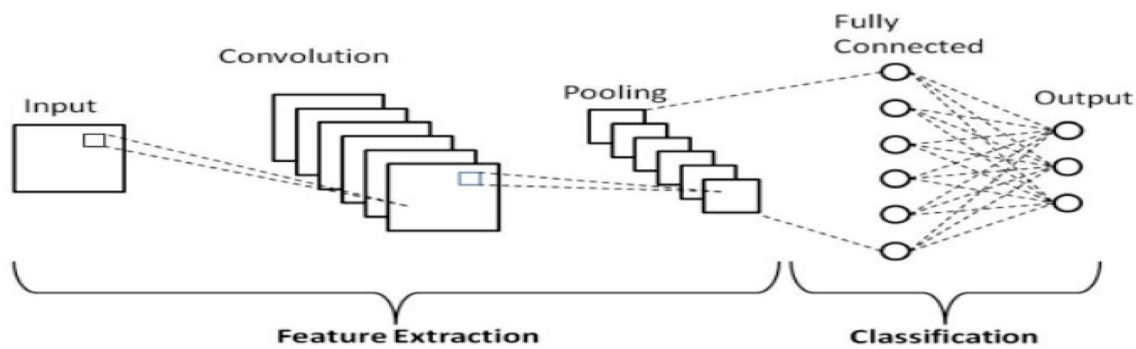


**Figure 3.4 :** Data Augmentation

### 3.2.4 CNN (Convolution Neural Network)

A Convolutional Neural Network (CNN) is a deep learning algorithm designed to automatically learn and extract meaningful features from input data through a series of computational layers. Convolutional neural network (CNN, or ConvNet) is a class of artificial

neural network (ANN), most commonly applied to analyze images. CNNs are regularized versions of multilayer perceptrons. Multilayer perceptron usually means fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The network learns to optimize the filters (or kernels) through automated learning. Convolution process is used during the learning phase. Learning actually means finding the right weights and biases and adjusting the weights and biases.



**Figure 3.5 : Convolution Neural Network**

### 3.2.5 CNN (Convolution Neural Network) Architecture

Our classifier is a convolutional neural network (CNN) consisting of

- 3 convolutional layers, each followed by a batch normalization and max pooling layer
- A fully connected layer
- An output layer with a single node, outputting a probability of inedible class

Convolution layer applies 32 filters of size 3x3 to the input images with a valid padding. The activation function used is ReLU (Rectified Linear Unit). The input shape is (64, 64, 3), indicating that the input images are 64x64 pixels with 3 color channels (RGB).

Batch normalization is applied to normalize the activations of the previous layer, helping with training stability and accelerating convergence. Max pooling layer performs max pooling with a pool size of 2x2 and a stride of 2. Max pooling reduces the spatial dimensions of the input by taking the maximum value within each pooling window.

Two more convolutional blocks with increasing filter sizes (64 and 128) and max pooling layers follow the first layer.

Flatten layer flattens the multi-dimensional output from the previous layer into a 1D vector, preparing it for the fully connected layers. These layers are fully connected neural layers. The first dense layer has 128 units with ReLU activation, followed by a dropout layer that randomly sets 10% of the inputs to 0 during training to prevent overfitting. The second dense layer has 64 units with ReLU activation, followed by another dropout layer. This is the final dense layer with a single unit and sigmoid activation. It outputs the predicted probability of the input belonging to the negative class. Each step contributes to the construction of the CNN model for image classification, progressively extracting features from the input images and making predictions based on those features.

The architecture of our model is visualized in the picture below. The architecture was derived by a combination of examples and recommendations and our own experimentation.



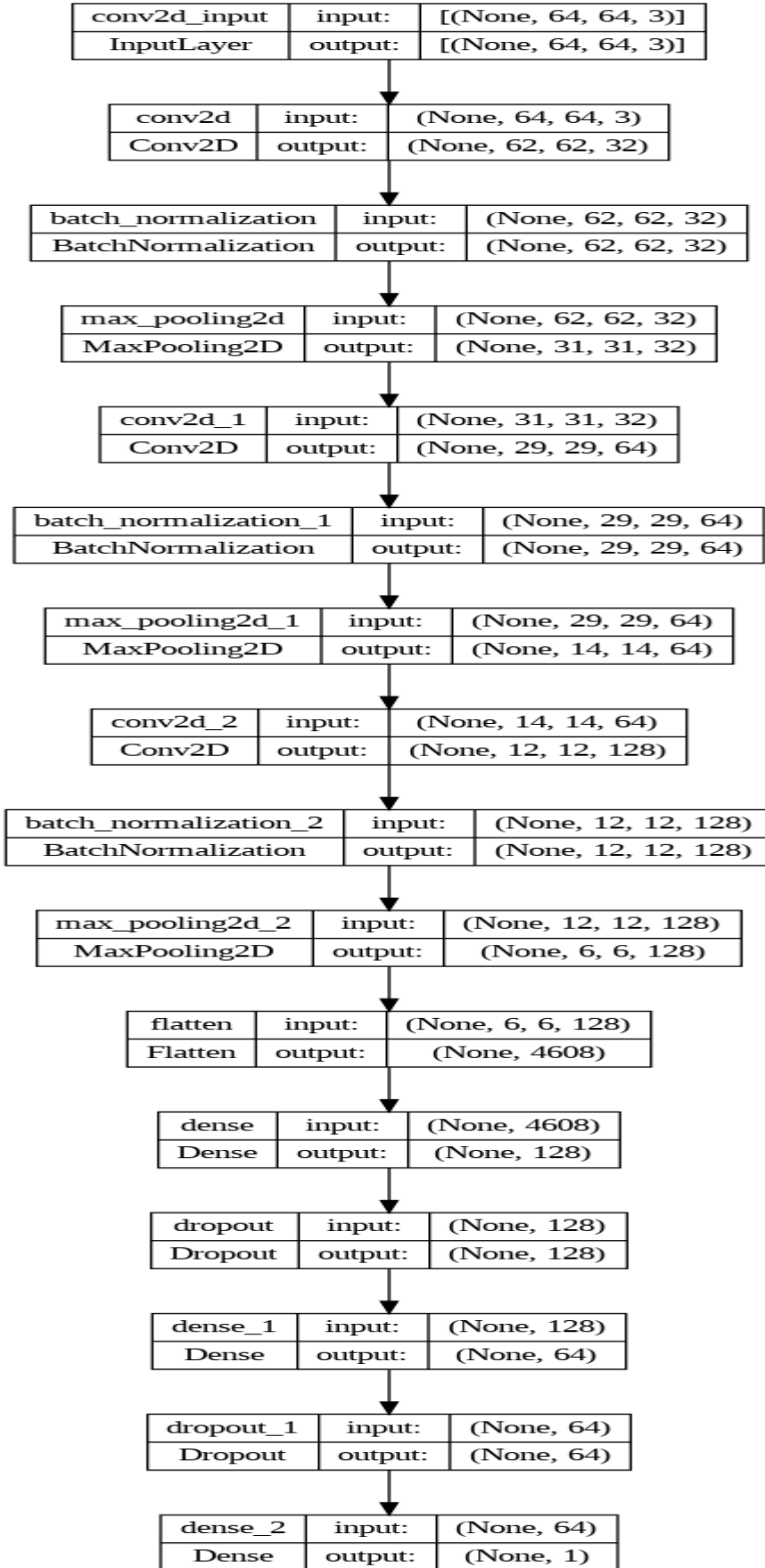


Figure 3.6 : CNN Architecture

### 3.2.5.1 Convolutional Layer

Convolutional layers are responsible for extracting features from the input data. Each convolutional layer consists of multiple filters (also known as kernels) that slide over the input data, performing a convolution operation. The output of this operation is a set of feature maps that represent different learned features. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter. The convolution layer in CNN passes the result to the next layer once applying the convolution operation in the input. The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image. Example of convolution operation is edge detection. Suppose the size of input image is  $n \times n$  and filter used is  $f \times f$  then the resulting output due to convolution will be  $n-f+1$ .

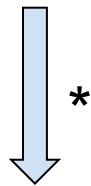
The depths of our 3 convolutional layers are 32, 64, and 128 respectively. This means that in the first layer, the convolution operation is performed separately by 32 different kernels each of size  $3 \times 3$ . The input to the first layer is an image of size  $(64, 64)$  with 3 channels producing a feature map of size  $(62, 62, 32)$ . In the second and third convolutional layers, the operation is performed 64 and 128 times respectively. The input to second layer is a feature map of size  $(31, 31, 32)$  producing the output feature map of size  $(29, 29, 64)$ . The input to third layer is a feature map of size  $(14, 14, 64)$  producing the output feature map of size  $(12, 12, 128)$ .

Examples of vertical edge detection can be illustrated through the given figure.



3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

**6\*6** Grey Scale Image



1	0	-1
1	0	-1
1	0	-1

**3\*3** Filter



-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

**4\*4** Final Image

**Figure 3.7 :** A 6x6x1 image convolved with a 3x3x1 kernel to become a 4x4x1 convolved feature.

### 3.2.5.2 Batch Normalization Layer

Each convolutional layer is followed by a batch normalization layer. Batch Normalization is a technique commonly used in deep neural networks, including Convolutional Neural Networks (CNNs), to improve training stability and speed up convergence. It helps address the internal covariate shift problem by normalizing the activations of a layer. In a CNN, the output activations of each layer are computed based on the input data and the learned weights. The distribution of these activations can change as the network parameters are updated during training. This phenomenon is known as the internal covariate shift. Batch Normalization normalizes the activations by subtracting the batch mean and dividing by the batch standard deviation. This normalization step brings the activations closer to zero mean and unit variance, which helps stabilize and speed up training.

### 3.2.5.3 Pooling Layer

Each convolutional layer is also followed by pooling. We used max pooling with pool size (2,2), which halves the input in both vertical and horizontal dimensions by outputting the maximum value of two disjoint consequent values in the feature map.

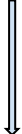
In a Convolutional Neural Network (CNN), the pooling layer is responsible for reducing the dimensions of the feature maps produced by the convolutional layers. It achieves this by applying a pooling operation to each slice or channel of the feature maps individually. The pooling operation replaces the output of the network at certain locations with a summary statistic (e.g., maximum value or average value) calculated from the nearby outputs. The purpose of this operation is to abstract the information, retaining the most important features while reducing the computational complexity and the number of parameters in the network. Pooling prevents the overfitting problem so that models don't learn in the complex way. If the input is of size  $n \times n$  and pool size is  $m \times m$  and stride of  $s$  then the resulting output will be  $(n-m)/s + 1$ .

Types of pooling operation are max pooling and average pooling. The choice of pooling operation may vary depending on the desired properties and objectives of the network, specific task and dataset.

The given example shows the max pooling operation of  $4 \times 4$  inputs with  $2 \times 2$  pool size and 2 strides.

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2

4\*4 Images


 2 X 2 Pooling  
matrix and Stride  
size 2

9	2
6	3

Result after max pooling

**Figure 3.8 :** Representation of Max Pooling layer

#### 3.2.5.4 Fully connected layer

A fully connected layer, also called a dense layer, is a fundamental component of neural networks. In this layer, every neuron is connected to every neuron in the previous layer, allowing information to flow freely throughout the network. Each neuron in the fully connected layer performs a weighted sum of its inputs, followed by an activation function to introduce non-linearity. This layer enables the network to learn complex relationships between the input data and the desired output. With its dense interconnections, the fully connected layer captures high-level representations and can be used for tasks such as classification or regression. However, the large number of parameters in fully connected layers can lead to increased computational complexity and a higher risk of overfitting, necessitating the use of regularization techniques to improve generalization.

In our model the first dense layer has 128 units with ReLU activation, followed by a dropout layer that randomly sets 10% of the inputs to 0 during training to prevent overfitting. The second dense layer has 64 units with ReLU activation, followed by another dropout layer. This is the final dense layer with a single unit and sigmoid activation. It outputs the predicted probability of the input belonging to the negative class. Each step contributes to the construction

of the CNN model for image classification, progressively extracting features from the input images and making predictions based on those features.

### 3.2.6 Activation function

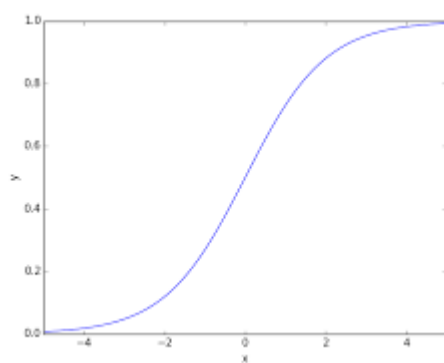
An activation function is a mathematical function that introduces non-linearity to neural networks. An activation function decides whether a neuron should be activated or not. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations. It determines the output of a neuron based on the weighted sum of its inputs. The activation function helps the neural network model to learn complex patterns and make predictions.

Two types of activation function have been used in our CNN model.

#### 1. Sigmoid Function

It is the activation function that returns the value between 0 and 1. The returned value of sigmoid function can be interpreted as probability. It can be represented as

$$f(x) = 1 / (1 + e^{(-x)})$$



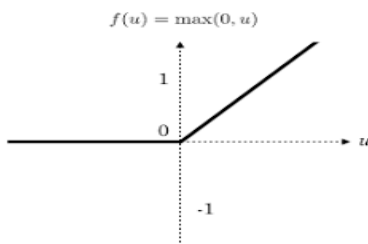
**Figure 3.9 : Sigmoid Function**

It has been used in the output layer with a single neuron. The output of the sigmoid function can be interpreted as the probability of input belonging to the inedible class

## 2. Re- LU Activation

Re- LU Activation (Rectified Linear Unit ) is an activation function which returns 0 if it receives any negative input, but for any positive value it returns that value back. It has been used in the convolutional layers to introduce non-linearity to the network, allowing CNNs to learn and model complex relationships in the data. Non-linear activation functions are crucial for capturing intricate patterns and enabling the network to approximate any arbitrary function. Mathematically it can be represented as  $f(u) = \max(0, u)$

Graphically it can be represented as



**Figure 3.10 : Re- LU Activation**

### 3.2.7 Optimization Algorithm

An optimization algorithm is a method or procedure used to find the optimal solution to a problem. It updates the parameters of a model during the training process. Its main goal is to minimize a loss function that quantifies the difference between the model's predictions and the actual target values. Optimization algorithm iteratively updates the model parameters based on the gradients of the loss function with respect to those parameters. The choice of the optimization algorithm depends on various factors such as the size and nature of the dataset, the model architecture, and computational constraints. We choose Adam optimizer as an optimization algorithm.

#### ➤ Adam Optimization Algorithm

The Adam (Adaptive Moment Estimation) combines concepts from both adaptive learning rate methods and momentum-based methods to efficiently update the model parameters during training. The algorithm adapts the learning rate for each parameter individually based on the estimates of the first and second moments of the gradients. The method is really efficient when working with large problems involving a lot of data or parameters.

### 3.2.8 Loss Function

In deep learning (DL), the loss function (also known as the cost function or objective function) is a mathematical function that quantifies the discrepancy between the predicted output of a model and the true target values. Discrepancy means inconsistency. In the context of deep learning, a discrepancy often refers to the difference between the predicted output of a model and the true target output. Loss function measures how well the model is performing and guides the optimization process to minimize this discrepancy. For our binary class classification we have used binary cross entropy as a loss function.

#### 3.2.8.1 Binary cross-entropy

Binary cross-entropy is a loss function used in binary classification to measure the dissimilarity between the predicted probability distribution and the true distribution when there are only two classes.

The output of the neural network is typically a single value between 0 and 1, representing the predicted probability of belonging to one of the two classes. The binary cross-entropy loss function quantifies the difference between the predicted probability and the true target. It is defined as:

$$\text{BCE} = - (\text{true} * \log(\text{predicted}) + (1 - \text{true}) * \log(1 - \text{predicted}))$$

### 3.2.9 Training the network

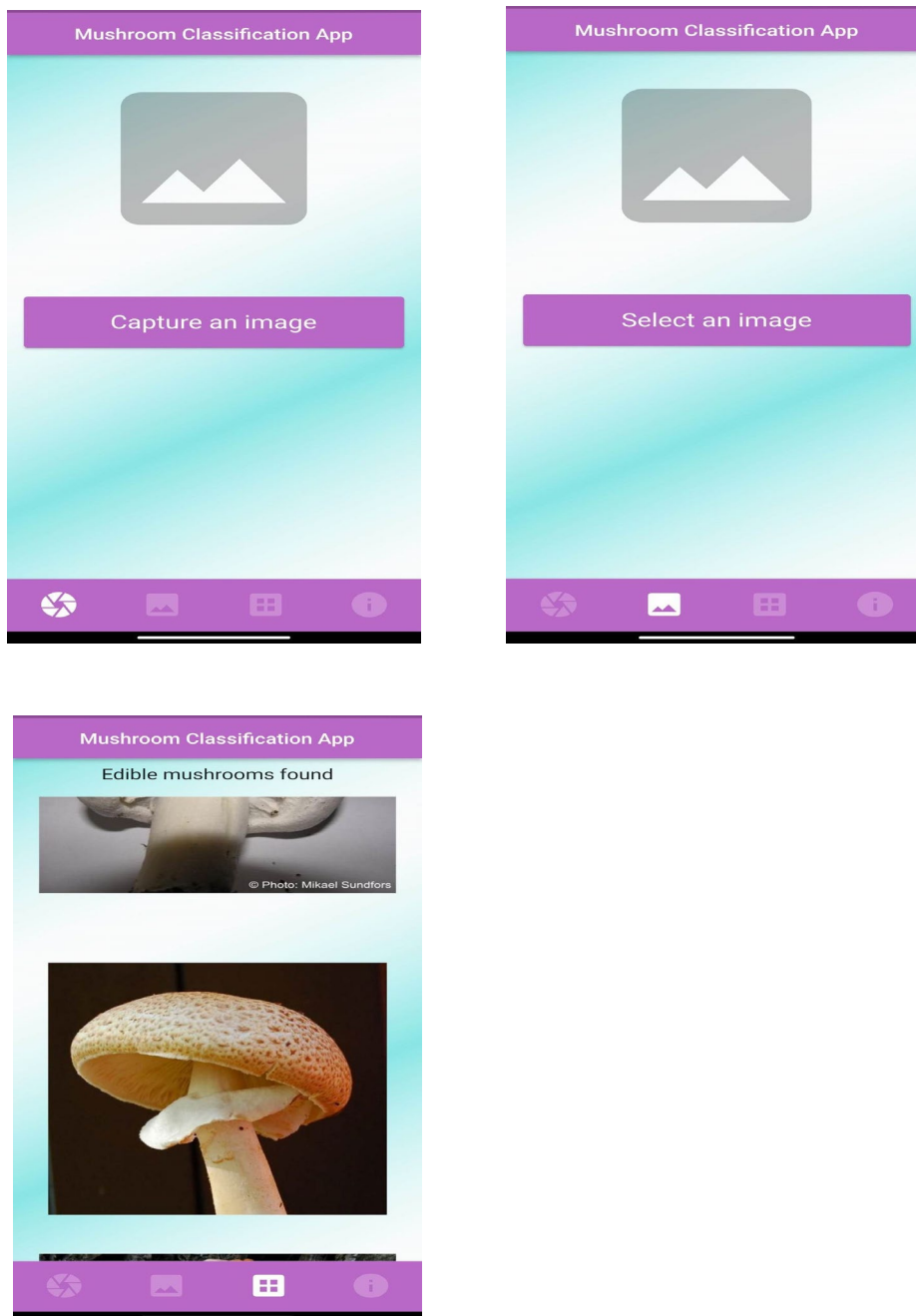
The image data was split to training and testing sets with a split ratio 8:2. The CNN model was then trained by feeding it 182 batches of augmented image data from the training set containing 128 images per batch. The network was trained on 50 epochs.

#### 3.2.10 Mobile application development

The smartphone application is built using flutter sdk. It consists of 4 screens namely capture image screen, select images screen, previous result screen and about screen.

In capture image screen we can capture images using our device camera and predict the edibility of mushroom. While in select image screen images can be selected from gallery and edibility can be checked. The CNN model is hosted in remote server. Images are sent to the model through Fast Api and edibility class is returned as response to be displayed in the

app. Successfully classified images are stored in firestore which can be viewed in previous result screen. About screen provides information about the developers.



**Figure 3.11 : Screenshoot of mobile UI**

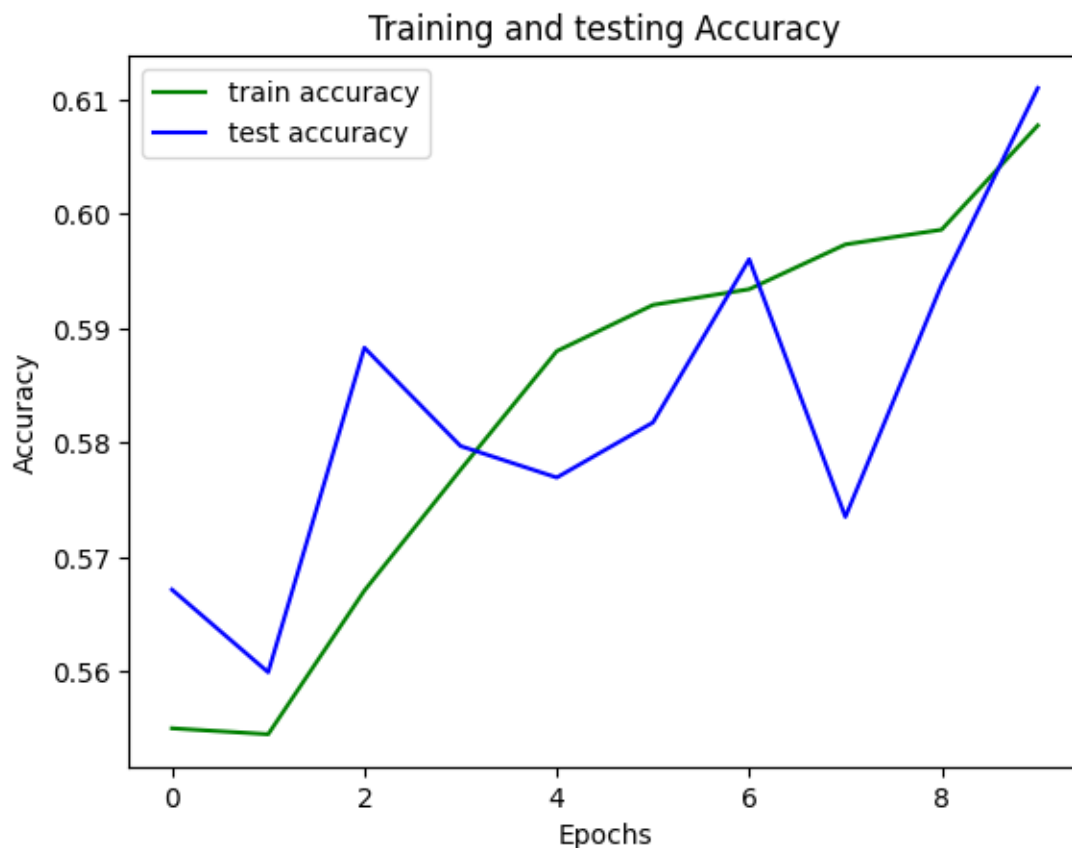
## CHAPTER 4: RESULT AND ANALYSIS

### 4.1 Experiment

The image data was splitted to training and testing sets with a ratio of 8:2. CNN model was trained and validated by feeding it small batches of augmented image data from the training and testing set, 128 images at a time. First the network was trained on 10 epochs and later it was trained on 50 epochs. Entirely the experiment was executed on python 3 google compute engine backend(GPU).

### 4.2 Result analysis

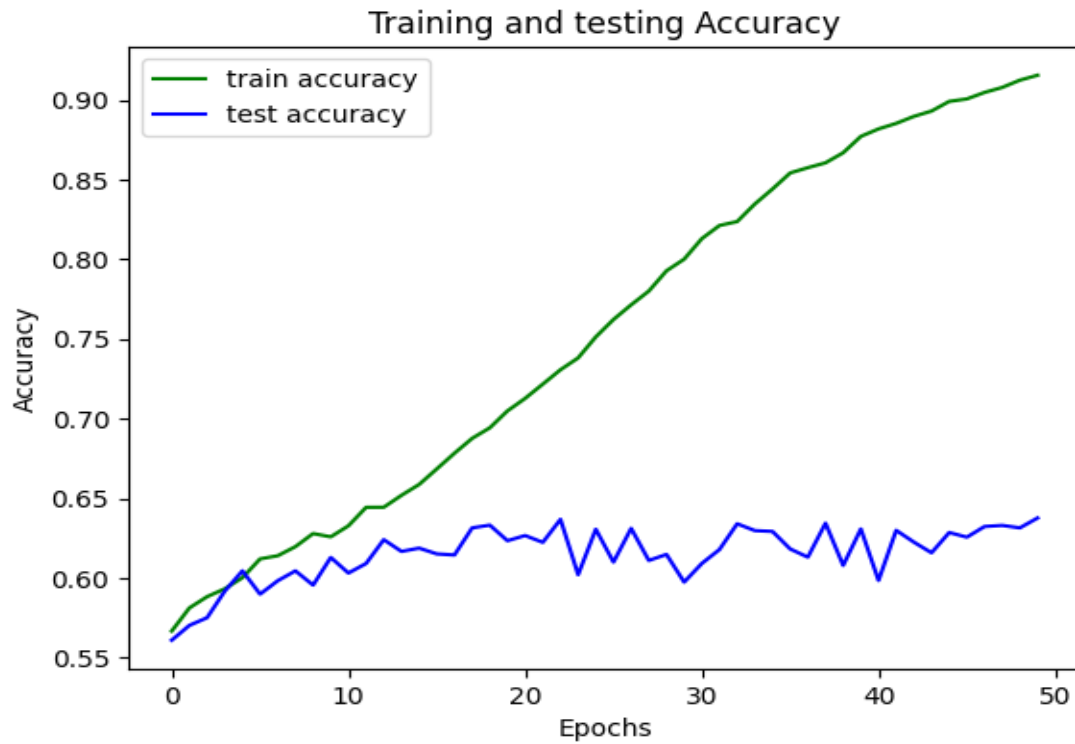
The graph below shows the training and testing accuracy for 10 and 50 epochs.



**Figure 4.1:** Training and testing accuracy plot for 10 epoches

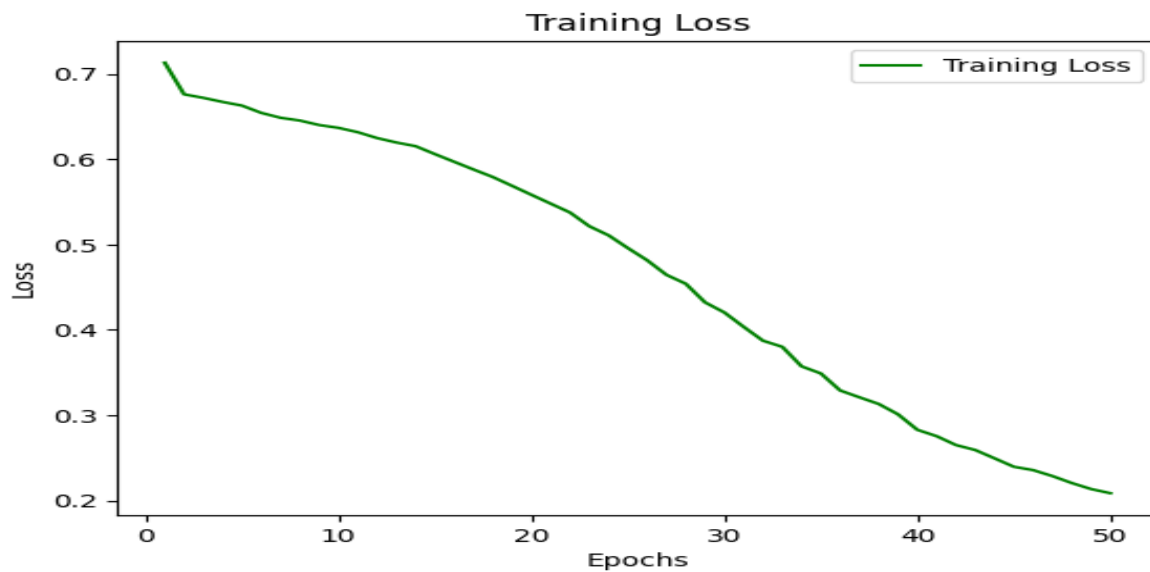
While training the model for 10 epochs at first, we obtained training accuracy as 60.2% and validation accuracy as 61.1%.



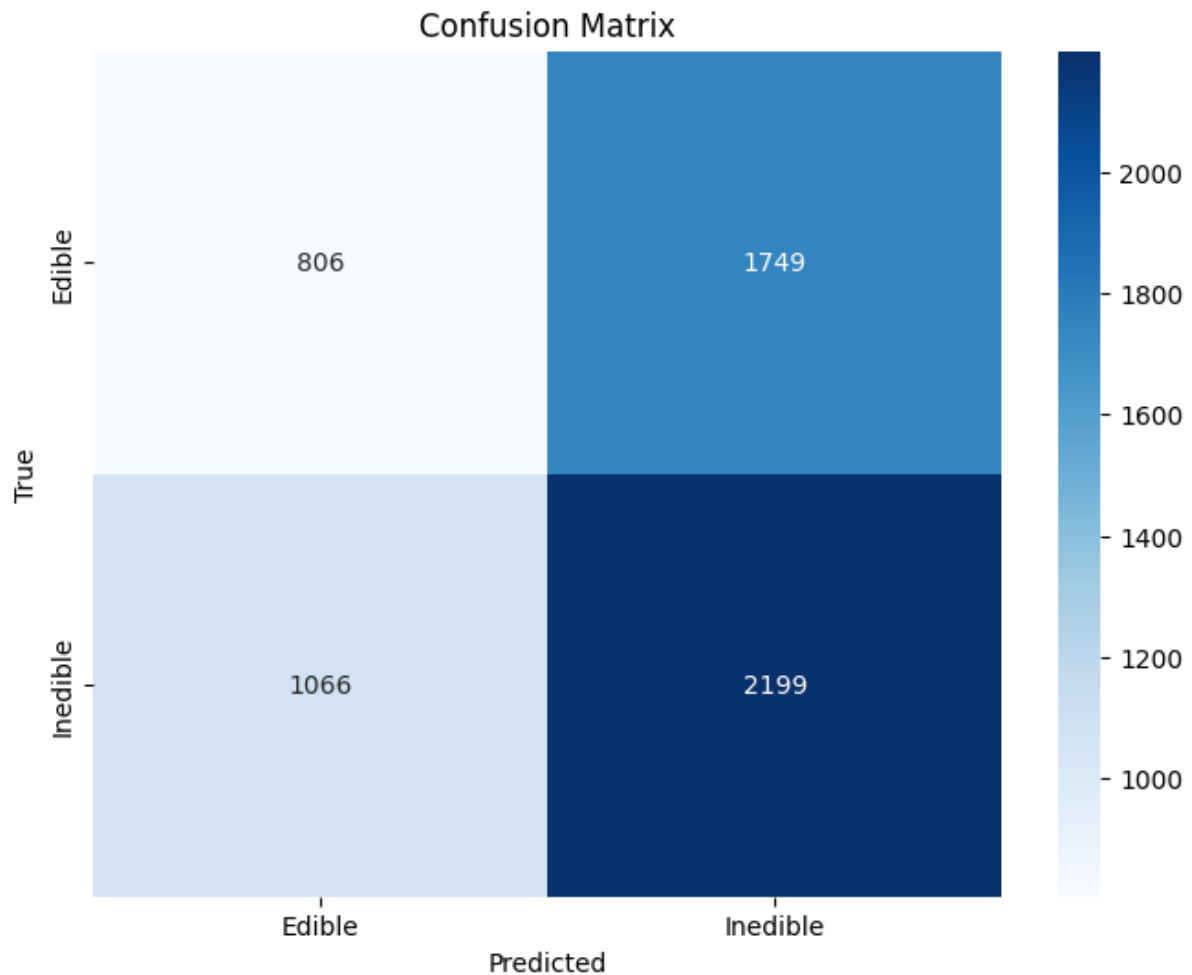


**Figure 4.2 :** Training and testing accuracy plot for 50 epoches

Batch normalization was added to each convolution layer and the model was retrained for 50 epochs. Then we obtained training accuracy as 91.65% and validation accuracy as 62.56%. We can see significant rise in training accuracy but not expected improvement in validation accuracy. It indicates overfitting in our model.



**Figure 4.3:** Training loss vs epochs



**Figure 4.4 :** Confusion matrix

#### 4.2.1 Evaluation Metrics

They provide insights about the performance of classification model and its ability to correctly classify positive and negative instances. Accuracy, precision, recall and F1 score for our model are shown below.

```
Accuracy: 0.5163230240549829
Precision: 0.5569908814589666
Recall: 0.6735068912710567
F1 Score: 0.6097324275613476
```

**Figure 4.5 :** Evaluation Metrics

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

It is the ratio of total correct prediction to total number of predictions.

$$\text{Precision} = \frac{TP}{TP+FP}$$

It is the ratio of correctly classified positive example to total number of predicted positive examples.

$$\text{Recall} = \frac{TP}{TP+FN}$$

It is the ratio of correctly classified positive example to total number of positive examples.

$$\text{F1 Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

It is the harmonic mean of precision and recall

TP = True Positive(Predicted positive and its true)

TN = True Negative(Predicted negative and its true)

FP = False Positive(predicted positive and its false)

FN = False Negative(predicted negative and its false)

### 4.3 Model predictions

Once the network has been trained, it can be used to make predictions. Our model outputs a single value describing the probability that an image belongs to inedible class.



```
[[0.36095282]]  
name:Suillus variegatus Variegated Bolete  
Actual edibility:edible  
Edible
```

Figure 4.6 : Model Prediction

## **CHAPTER 5: CONCLUSION**

To sum up, we created a CNN model consisting of 3 convolutional layers followed by a fully connected layer to classify mushroom images as edible and inedible. The model was trained and validated by feeding small batches of image data from the training and validation set. The network was trained on 50 epochs. In this way we obtained 91% training accuracy and 63% validation accuracy. The model has been remotely hosted and can be accessed through a mobile app to make prediction. The smartphone application provides a user-friendly interface for capturing the mushroom images in real time as well as inputting pre-saved mushroom images and checking the edibility class.

It is important to note that accurately classifying mushrooms as edible or inedible solely based on images can be challenging and potentially risky. Mushroom identification should not solely rely on image classification models. Expert knowledge, consultation with mushroom experts, and additional information like geographic location, habitat, and other characteristics should be considered for accurate identification and determination of edibility.

## REFERENCES

- [1] Lee, Jae Joong, et al. "Machine Learning-Based Classification of Mushrooms Using a Smartphone Application." *Applied Sciences* 12.22 (2022): 11685.
- [2] Ottom, Mohammad Ashraf, Noor Aldeen Alawad, and K. M. Nahar. "Classification of mushroom fungi using machine learning techniques." *International Journal of Advanced Trends in Computer Science and Engineering* 8.5 (2019): 2378-2385.
- [3] Zahan, Nusrat, et al. "A Deep Learning-Based Approach for Edible, Inedible and Poisonous Mushroom Classification." *2021 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD)*. IEEE, 2021.
- [4] Alkronz, Eyad Sameh, et al. "Prediction of whether mushroom is edible or poisonous using back-propagation neural network." (2019).
- [5] Chowdhury, Dilip Roy, and Subhashish Ojha. "An empirical study on mushroom disease diagnosis: a data mining approach." *International Research Journal of Engineering and Technology (IRJET)* 4.1 (2017): 529-534.