DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalgodu, Mysuru Road, Bengaluru-560 074

Report On

"AI-based Food Recognition for theVisually impaired"

Internship carried out in the

Department of Computer Science & Engineering

Under DBIT-SIC

Submitted by

NIKHIL M [1DB20IS094]

HARSHITA S H [1DB20CS047]

Under the guidance of

External Guide	**Internal Guide**
Mr. Prasad K	Dr. Venugeetha Y
Trainer	Professor, Coordinator – SIC
SIC	Dept. of CSE DBIT, Bengaluru

Held at

DBIT-SIC Laboratory

2022-2023

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Mysore Road, Bengaluru - 560074 www.dbit.co.in ph: +91-80-28437028/29/30 Fax: +91-80-28437031

Department of Computer Science & Engineering

Under DBIT-SIC

Ref # DBIT-SIC-AI / B1 / 2023 Date :

INTERNSHIP CERTIFICATE

TO WHOM IT MAY CONCERN

This is to certify that Mr. Nikhil M of Department of Information Science and Engineering bearing USN: 1DB20IS094 has successfully completed an internship program on Artificial Intelligence domain during 14th August 2023 to 9th September 2023 with a Capstone Project of 90 hours followed by a presentation.

Co-ordinator SIC **HoD_CSE** **Principal**

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Mysore Road, Bengaluru - 560074 www.dbit.co.in ph: +91-80-28437028/29/30 Fax: +91-80-28437031

Department of Computer Science & Engineering

Under DBIT-SIC

Ref # DBIT-SIC-AI / B1 / 2023 Date :

INTERNSHIP CERTIFICATE

TO WHOM IT MAY CONCERN

This is to certify that Ms. Harshita S H of Department of Computer Science and Engineering bearing USN: 1DB20CS047 has successfully completed an internship program on Artificial Intelligence domain during 14th August 2023 to 9th September 2023 with a Capstone Project of 90 hours followed by a presentation.

Co-ordinator SIC **HoD_CSE** **Principal**

ABSTRACT

This project presents an innovative food item classification system designed to assist visually challenged individuals in identifying various types of foods. The system utilizes the capabilities of deep learning through the implementation of two distinct approaches: a custom convolutional neural network (CNN) and the InceptionV3 pre-trained model.

The custom CNN architecture features multiple convolutional and pooling layers, as well as fully connected layers, enabling it to extract intricate image features. Extensive data augmentation techniques are employed to enhance the model's ability to generalize across diverse food items. Additionally, the project leverages the transfer learning approach to fine-tune the InceptionV3 model for food classification.

The integration of InceptionV3 results in a notable improvement in classification accuracy and loss**.** One of the key features of this system is its accessibility component, which provides auditory output for visually challenged users. Upon analyzing an input image, the system utilizes text-to-speech functionality to verbalize the identified food item, ensuring that users can independently and confidently identify the foods in their surroundings.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter-1

INTRODUCTION

An "AI-based Food Recognition for Visually impaired" project is a cutting-edge application of technology designed to assist individuals with visual impairments in identifying and recognizing different types of food. The main objective is to provide real-time audio descriptions of food items, empowering visually challenged individuals to make informed choices about their meals without relying on sight.

In this innovative project, computer vision and machine learning technologies are harnessed to process images of food and convert the results into audio descriptions. By uploading food images, users can access detailed audio information, enhancing their dining experiences and aiding those with visual impairments .This technology not only enhances the independence of visually impaired individuals in daily food-related activities but also promotes healthier dietary choices and inclusivity, ensuring they can participate more fully in social and culinary experiences.

In a world that heavily relies on visual cues for food selection and presentation, this project serves as a significant breakthrough for the visually challenged community. With its real-time audio-based food recognition, it not only offers practical assistance but also fosters a sense of empowerment and autonomy. The system's core features include seamless image capture, robust image preprocessing, and a powerful food recognition model based on deep learning. This model is capable of identifying a wide range of foods, spanning various cuisines . The recognized food items are then transformed into clear and articulate audio descriptions using text-to-speech technology.

This system is designed to bridge the accessibility gap for visually impaired individuals and provide a seamless, user-friendly experience. By analyzing food images, our platform offers audio descriptions, making it easier for users to identify and appreciate different dishes while dining out or ordering takeout. This technology not only promotes inclusivity but also enhances the overall dining experience for everyone, ensuring that food choices are informed and enjoyable.

1.1 Existing System

- The existing systems for food recognition catering to visually challenged individuals are primarily characterized by manual assistance and specialized assistive devices. Historically, visually challenged individuals have relied on the assistance of care givers or restaurant staff when identifying and selecting food items, which can be time-consuming and dependent on human availability. Some solutions involve the use of barcode scanners and mobile apps, allowing users to take pictures for text-based descriptions of packaged foods.
- Manual Assistance: Visually challenged individuals historically rely on manual assistance from caregivers or restaurant staff for food identification and selection.
- Specialized Assistive Devices: Some solutions involve the use of specialized assistive devices, such as barcode scanners, for identifying packaged foods.
- Custom Devices: Custom devices, including handheld scanners and smart glasses, have been developed for food recognition. While they offer food recognition capabilities, they can be costly and come with a learning curve for users.

  Demerits are:

  - Limited Coverage: Existing solutions may have limited capabilities in recognizing a wide variety of food items, especially those prepared at home or in smaller restaurants.
  - Complex Dishes: Identifying complex dishes with multiple ingredients and preparations can be challenging for existing systems.
  - Cost and Accessibility: Specialized assistive devices and custom solutions can be expensive, potentially limiting access for individuals with limited financial resources.

1.2 Problem Statement

Visually impaired individuals encounter substantial obstacles when it comes to autonomously identifying and choosing their meals, whether at home or in public dining settings. Current solutions, including manual assistance and specialized devices, present shortcomings in terms of their ability to recognize a broad range of food items, provide real-time feedback, and remain cost-effective. Consequently, there is an urgent demand for an innovative and inclusive system that harnesses the capabilities of computer vision and audio technology to enable immediate real-time food recognition for visually impaired individuals, promoting their independence and enhancing their dining experiences.

1.3 Objectives

The "AI-Based Food Recognition for Visually Impaired" project is driven by a comprehensive set of objectives aimed at enhancing the lives of visually impaired individuals through innovative technology. The project seeks to develop a real-time food recognition system that leverages cutting-edge AI and deep learning models to provide instantaneous identification and classification of a wide range of food items. By achieving a high level of accuracy, the system ensures that visually impaired users receive reliable and precise information about the food before them. Crucially, the recognized food items will be converted into clear and articulate audio descriptions, facilitating easy comprehension for users.

The project seeks to provide immediate and precise audio descriptions of recognized food items, eliminating delays in feedback. This objective directly addresses the issue of delayed responses found in existing systems, ensuring that visually challenged users can enjoy a seamless and timely dining experience.

Develop an image-based food recognition system using the InceptionV3 model and CNN architecture to accurately identify and classify various food items and dishes based on static images and Implement a feedback mechanism to gather user input for continuous model improvement and system enhancements.

Chapter2

LITERATURE REVIEW

Sl no.	References	Databases used	Deep Learning approach used	Abstract	Contribution towards classification and accuracy
1.	Gurbakash Phonsa (2021)	cifar-10 dataset	Convolutional neural network(CNN)	Content Based Image Retrieval Technique(CBIR) classification by usingCNN.	CNN to achieve 94% accuracy
2.	Mingyuan Xin & Yong Wang (2019)	MNIST and CIFAR-10	Support vector machine (SVM) and CNN	M3CE-CEc enhances MNIST, CIFAR-10 combining cross entropy and M3CE.	Model to achive accuracy of 85%.
3.	NehaSharma, Vibhor Jain (201 8)	ImageNet, CIFAR10, CIFAR100.	AlexNet, GoogLeNet, ResNet50.	CNN performance on object recognition.	Achieve accuracy of 92.04%.
4.	Lihua Luo (2021)	Conf. Ser. Volume 2803	CNN and deep belief network.	Importance of image classification algorithms.	Analyzing image classification algorithms impact.
5.	Farhana Sultana,Abu Sufian(2018)	Conf. ICRCICN.2018. 871871	CNN architectures LeNet-5 and SENet model.	CNN Classification Advancements	Advances in Classification with CNN.

Chapter-3

PROPOSED SYSTEM

3.1Requirements

3.1.1 Hardware Requirements

- Processor : Intel Core 3 and above
- RAM: 8GB
- Operating System: Windows 98

3.1.2 Software Requirements

- Programming language : Python Version 3.6 and later
- IDE: Google Colab
- Front End Tool: Streamlit , CSS
- Libraries Used:
    1. Streamlit
    2. Numpy
    3. Matplotlib
    4. TenserFlow
    5. gTTS (Google Text-to-Speech)
    6. Pillow (PIL)

3.2 Library

Streamlit : Streamlit is a Python library used to create web applications for data science and machine learning. Streamlit is an open-sourcePython library that simplifies the creation of web applications with minimal effort.

gTTS (Google Text-to-Speech) or pyttsx3:gTTS is a Python library and CLI tool that interfaces with Google Translate's text-to-speech API. pyttsx3 is a cross-platform text-to-speech conversion library. With gTTS, you can easily add speech functionality to your Python scripts, allowing you to create audio files from any text.

Matplotlib: Matplotlib is a popular Python library for creating static, animated, or interactive visualizations. Matplotlib is a popular Python library for creating static, animated, or interactive visualizations.

TenserFlow : TensorFlow is an open-source machine learning framework developed by Google that is widely used for building and training machine learning and deep learning models. TensorFlow is an open-source machine learning framework developed by Google that is widely used for building and training machine learning and deep learning models.

Pillow (PIL): Pillow is a Python Imaging Library that allows you to open, manipulate, and save image files. It's useful for image processing and manipulation. incorporates lightweight image processing tools that aids in editing, creating and saving images.

NumPy: NumPy, short for "Numerical Python," is a fundamental library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with an extensive collection of mathematical functions to operate on these arrays.

3.3 DATASET

The food recognition dataset is dynamically retrieved from a live image feed or online database based on user input. Unlike conventional datasets that are sourced from static files, this dataset relies on real-time image capture or access to an online repository of food-related images. The system leverages specialized libraries and APIs to capture or access images dynamically and conduct food recognition tasks in real-time. It continuously updates the dataset as new images become available, ensuring that it reflects the latest and most relevant visual data for food recognition. This dynamic nature of the dataset allows for more accurate and up-to-date recognition of various food items and culinary creations.

we present a dataset of various food items that can be used for analysis or reference. This dataset includes a wide range of dishes from different cuisines. Please find the list of food items below:

Apple Pie	Baby Back Ribs	Baklava
Beef Carpaccio	Beef Tartare	Beet Salad
Beignets	Bibimbap	Bread Pudding
Breakfast Burrito	Bruschetta	Caesar Salad
Cannoli	Caprese Salad	Carrot Cake
Ceviche	Cheese Plate	Cheesecake
Chicken Curry	Chicken Quesadilla	Chicken Wings
Chocolate Cake	Chocolate Mousse	Churros
Clam Chowder	Club Sandwich	Crab Cakes
Creme Brulee	Croque Madame	Cupcakes
Deviled Eggs	Donuts	Dumplings
Edamame	Eggs Benedict	Escargots
Falafel	Filet Mignon	Fish and Chips
Foie Gras	French Fries	French Onion Soup
French Toast	Fried Calamari	Fried Rice
Frozen Yogurt	Garlic Bread	Gnocchi
Greek Salad	Grilled Cheese Sandwich	Grilled Salmon
Guacamole	Gyoza	Hamburger
Hot and Sour Soup	Hot Dog	Huevos Rancheros
Hummus	Ice Cream	Lasagna
Lobster Bisque	Lobster Roll Sandwich	Macaroni and Cheese
Macarons	Miso Soup	Mussels
Nachos	Omelette	Onion Rings

Oysters	Pad Thai	Paella
Pancakes	Panna Cotta	Peking Duck
Pho	Pizza	Pork Chop
Poutine	Prime Rib	Pulled Pork Sandwich
Ramen	Ravioli	Red Velvet Cake
Risotto	Samosa	Sashimi
Scallops	Seaweed Salad	Shrimp and Grits
Spaghetti Bolognese	Spaghetti Carbonara	Spring Rolls
Steak	Strawberry Shortcake	Sushi
Tacos	Takoyaki	Tiramisu
Tuna	Tartare	Waffles

3.3.1 Understanding of Datasets

1. **Images:** The heart of the dataset comprises a vast collection of images, each depicting various food items. These images vary in terms of content, composition, quality and resolution close.
2. **Metadata:** Metadata contains supplemental information associated with each image in the dataset. Key elements of metadata may include:
   - **Labels or Food Classes:** Metadata should specify which of the 101 food classes each image belongs to.
   - **Image File Information:** Metadata can include details about the file, such as file paths, image file formats.
3. **Training Data:** The training data subset is dedicated to training machine learning models. It includes a substantial portion of the dataset's images paired with their corresponding food class labels.
4. **Test Data:** The test data subset is reserved for evaluating your trained models. It contains images that the model hasn't seen during training.
5. **Class Distribution:** To comprehend the dataset thoroughly, it's vital to examine the distribution of images across the 101 food classes.

3.4 Algorithm

1. **Data Collection:** The algorithm begins with a dataset of food-related images, including a diverse range of food items from the 101 specified food classes.
2. **Preprocessing:** The images are preprocessed to ensure uniformity and optimal input for the algorithm. Common preprocessing steps include resizing images to a consistent resolution, normalizing pixel values, and possibly applying data augmentation techniques to increase the dataset's diversity.
3. **Feature Extraction:** Convolutional Neural Networks (CNNs) are often used for feature extraction in image recognition tasks. The algorithm employs Custom model which uses CNN architecture as a feature extractor.
4. **Model Fine-Tuning:** To adapt the pre-trained model for food recognition, the final classification layers are adjusted.
5. **Training:** The algorithm trains the model using the preprocessed training data, where ach image is associated with its corresponding food class label.
6. **Evaluation:** After training, the algorithm evaluates the model's performance using the test data. It measures metrics such as accuracy, precision, recall, and F1-score to assess the model's ability to recognize various food items.
7. **Prediction**: With a trained model, the algorithm can make predictions on new, unseen food images.
8. **Real-Time Recognition (if applicable):** If designed for real-time recognition, the algorithm continuously captures or accesses food images and feeds them to the model for immediate classification. This mode enables dynamic recognition of food items as they are presented in real life
9. **Post-Processing and Visualization**: The algorithm may include post-processing steps to enhance the interpretability of results. This can involve generating text descriptions of recognized food items, along with any additional information, and possibly converting text descriptions to audio for user convenience

3.5Coding

The script utilizes Streamlit to create a simple web interface for uploading food images and employs a pre-trained TensorFlow model to predict the food item, displaying both the result and generating a corresponding audio description.

```
!pip install streamlit -q
!pip install gTTS
%%writefile app.py
import numpy as np
import streamlit as st
import tensorflow as tf
from PIL import Image, ImageOps
from gtts import gTTS
def import_and_predict(image_data, model):
    size = (75, 75)
    image = ImageOps.fit(image_data, size, method=Image.LANCZOS)
    image = image.convert('RGB')
    image = np.asarray(image)
    image = (image.astype(np.float32) / 255.0)
    img_reshape = image[np.newaxis, ...]
    prediction = model.predict(img_reshape)
    return prediction
model = tf.keras.models.load_model('bestmodel_3class.hdf5')
st.write("""
         # Food Item Prediction
         """
         )
st.write("This is a simple image classification web app to predict food item names")
file = st.file_uploader("Please upload an image file", type=["jpg", "png"])
if file is None:
    st.text("You haven't uploaded an image file")

else:
    image = Image.open(file)
    st.image(image, use_column_width=True)
    prediction = import_and_predict(image, model
    food_classes = ['samosa','pizza','omelette',’french_fries’,’icecream’,’spring_rolls’]
    # Display the prediction
    food_class = food_classes[np.argmax(prediction)]
    st.write(f"It is a {food_class}!")
# Text-to-speech
    text_to_speech = f"It is a {food_class}!"
    tts = gTTS(text=text_to_speech, lang='en')
     tts.save("output.mp3")
     # Play audio
    st.audio("output.mp3", format="audio/mp3", start_time=0)
```

#Add all the imports

This code snippet includes multiple Python import statements that load various libraries and modules, such as TensorFlow, Keras, OpenCV, and more, to support deep learning, image processing, and data manipulation tasks in a project.

```
from __future__ import absolute_import, division, print_function
import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras import regularizers
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.layers import GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, CSVLogger
from tensorflow import keras
from tensorflow.keras import models
from tensorflow.keras.applications.inception_v3 import preprocess_input
import cv2
import os
import random
import collections
from collections import defaultdict
from shutil import copy
from shutil import copytree, rmtree
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as img
```

# Check TF version and whether GPU is enabled

```
print(tf.__version__)
print(tf.test.gpu_device_name())
```

# Clone tensorflow/examples repo which has images to evaluate trained model

```
!git clone https://github.com/tensorflow/examples.git
# Helper function to download data and extract
def get_data_extract():
  if "food-101" in os.listdir():
    print("Dataset already exists")
  else:
    tf.keras.utils.get_file(
    'food-101.tar.gz',
    'http://data.vision.ee.ethz.ch/cvl/food-101.tar.gz',
      cache_subdir='/content',
      extract=True,
   archive_format='tar',
     cache_dir=None
     )
     print("Dataset downloaded and extracted!")
```

# Download data and extract it to folder

```
get_data_extract()
```

# Check the extracted dataset folder

```
os.listdir('food-101')
```

# Visualize the data, showing one image per class from 101 classes

```
rows = 17
cols = 6
fig, ax = plt.subplots(rows, cols, figsize=(25,25))
fig.suptitle("Showing one random image from each class", y=1.05, fontsize=24) # Adding  y=1.05,
fontsize=24 helped me fix the suptitle overlapping with axes issue
data_dir = "food-101/images/"
foods_sorted = sorted(os.listdir(data_dir))
food_id = 0
for i in range(rows):
  for j in range(cols):
    try:
      food_selected = foods_sorted[food_id]
      food_id += 1
    except:
break
    food_selected_images = os.listdir(os.path.join(data_dir,food_selected)) # returns the list of all files
present in each food category

img = plt.imread(os.path.join(data_dir,food_selected, food_selected_random))

ax[i][j].imshow(img)
    ax[i][j].set_title(food_selected, pad = 10)

plt.setp(ax, xticks=[],yticks=[])
plt.tight_layout()
# https://matplotlib.org/users/tight_layout_guide.html
```

# Helper method to split dataset into train and test folders

```
def prepare_data(filepath, src,dest):
  classes_images = defaultdict(list)
  with open(filepath, 'r') as txt:
      paths = [read.strip() for read in txt.readlines()]
      for p in paths:
        food = p.split('/')
        classes_images[food[0]].append(food[1] + '.jpg')
for food in classes_images.keys():
    print("\nCopying images into ",food)
    if not os.path.exists(os.path.join(dest,food)):
      os.makedirs(os.path.join(dest,food))
    for i in classes_images[food]:
      copy(os.path.join(src,food,i), os.path.join(dest,food,i))
  print("Copying Done!")
```

# Prepare train dataset by copying images from food-101/images to food-101/train using the file train.txt

```
print("Creating train data...")
prepare_data('food-101/meta/train.txt', 'food-101/images', 'food-101/train')
```

# Check how many files are in the train folder

```
train_files = sum([len(files) for i, j, files in os.walk("food-101/train")])
print("Total number of samples in train folder")
print(train_files)
```

# Check how many files are in the test folder

```
test_files = sum([len(files) for i, j, files in os.walk("food-101/test")])
print("Total number of samples in test folder")
print(test_files)
```

# List of all 101 types of foods(sorted alphabetically)

```
foods_sorted
```

# Helper method to create train_mini and test_mini data samples

```
def dataset_mini(food_list, src, dest):
  if os.path.exists(dest):
    rmtree(dest) # removing dataset_mini(if it already exists) folders so that we will have only the classes
that we want
  os.makedirs(dest)
  for food_item in food_list :
    print("Copying images into",food_item)
copytree(os.path.join(src,food_item), os.path.join(dest,food_item))
```

# picking 3 food items and generating separate data folders for the same

```
food_list = ['samosa','pizza','omelette']
src_train = 'food-101/train'
dest_train = 'food-101/train_mini'
src_test = 'food-101/test'
dest_test = 'food-101/test_mini'
print("Creating train data folder with new classes")
dataset_mini(food_list, src_train, dest_train)
print("Total number of samples in train folder")

train_files = sum([len(files) for i, j, files in os.walk("food-101/train_mini")])
print(train_files)
print("Creating test data folder with new classes")
dataset_mini(food_list, src_test, dest_test)
print("Total number of samples in test folder")
test_files = sum([len(files) for i, j, files in os.walk("food-101/test_mini")])
```

```
print(test_files)
```

#Fine tune Custom model using Food 101 dataset

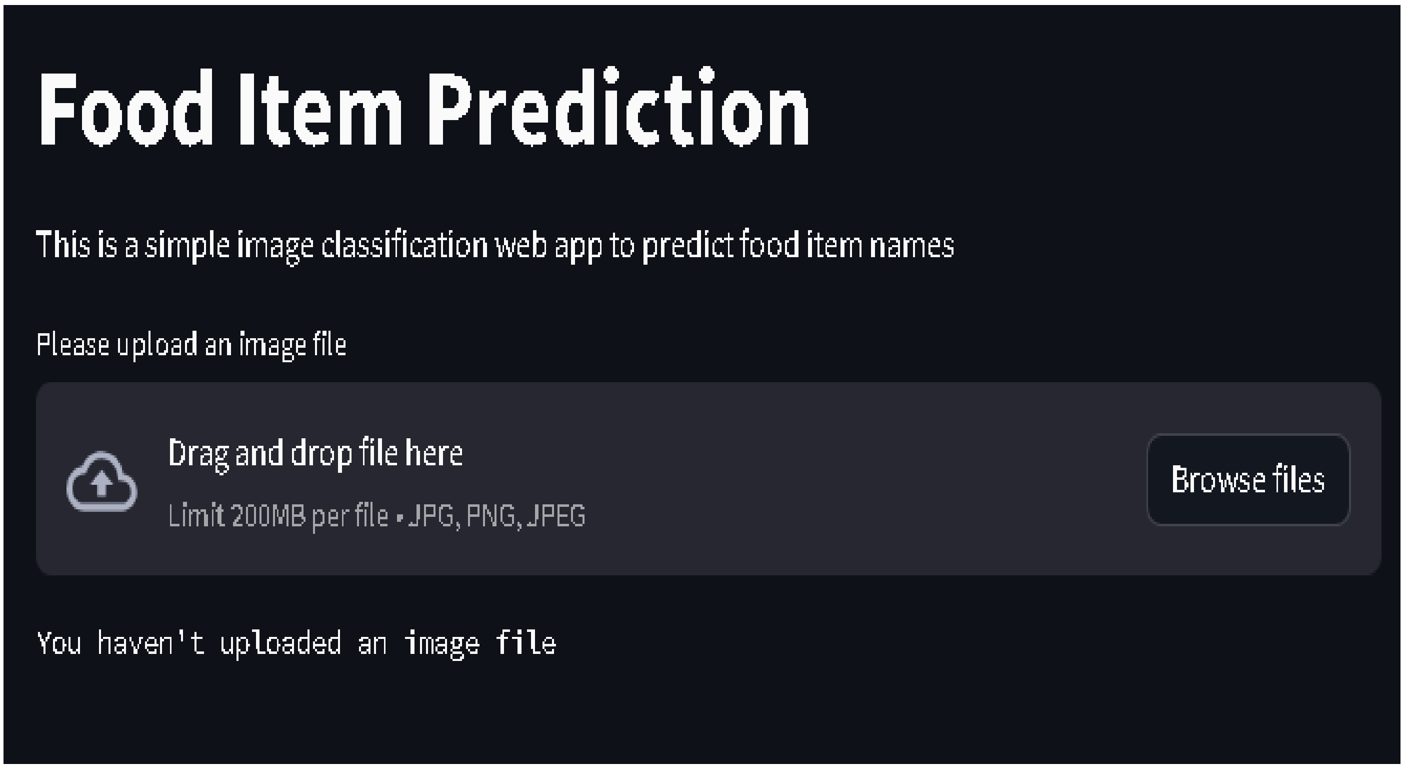
```
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Dropout
from keras.preprocessing.image import ImageDataGenerator
```

# Define the model

```
model = Sequential()
model.add(Conv2D(32, 3, padding="same", activation="relu", input_shape=(224, 224, 3)))
model.add(MaxPool2D())

model.add(Conv2D(32, 3, padding="same", activation="relu"))
model.add(MaxPool2D())

model.add(Conv2D(64, 3, padding="same", activation="relu"))
model.add(MaxPool2D())
model.add(Dropout(0.4))

model.add(Flatten())
model.add(Dense(128, activation="relu"))

model.add(Dense(6 activation="softmax"))  # Assuming 3 classes in your dataset

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Data Augmentation
train_datagen = ImageDataGenerator(
    rescale=1.0 / 255.0,
    rotation_range=10,
    zoom_range=0.2,
    width_shift_range=0.1,
    height_shift_range=0.1
)

test_datagen = ImageDataGenerator(rescale=1.0 / 255.0)
train_generator = train_datagen.flow_from_directory(
    path_to_train_data,
    target_size=(224, 224),
    batch_size=16,
    class_mode='categorical'
)

valid_generator = test_datagen.flow_from_directory(
    path_to_valid_data,
    target_size=(224, 224),
    batch_size=16,
    class_mode='categorical' )
```

```
# Train the custom model on your dataset

history=model.fit(
    train_generator,
    steps_per_epoch=len(train_generator),
    epochs=20,
    validation_data=valid_generator,
    validation_steps=len(valid_generator)
)

# Save the trained model
model.save('custom_model.h5')

#Visualisation
import matplotlib.pyplot as plt
# Get the training history
training_loss = model.history.history['loss']
validation_loss = model.history.history['val_loss']
training_accuracy = model.history.history['accuracy']
validation_accuracy = model.history.history['val_accuracy']
# Number of epochs
epochs = range(1, len(training_loss) + 1)
# Plot training and validation loss
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(epochs, training_loss, 'r', label='Training Loss')
plt.plot(epochs, validation_loss, 'b', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
# Plot training and validation accuracy
plt.subplot(1, 2, 2)
plt.plot(epochs, training_accuracy, 'r', label='Training Accuracy')
plt.plot(epochs, validation_accuracy, 'b', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

3.6 Result and Snapshots


Food Item Prediction

This is a simple image classification web app to predict food item names

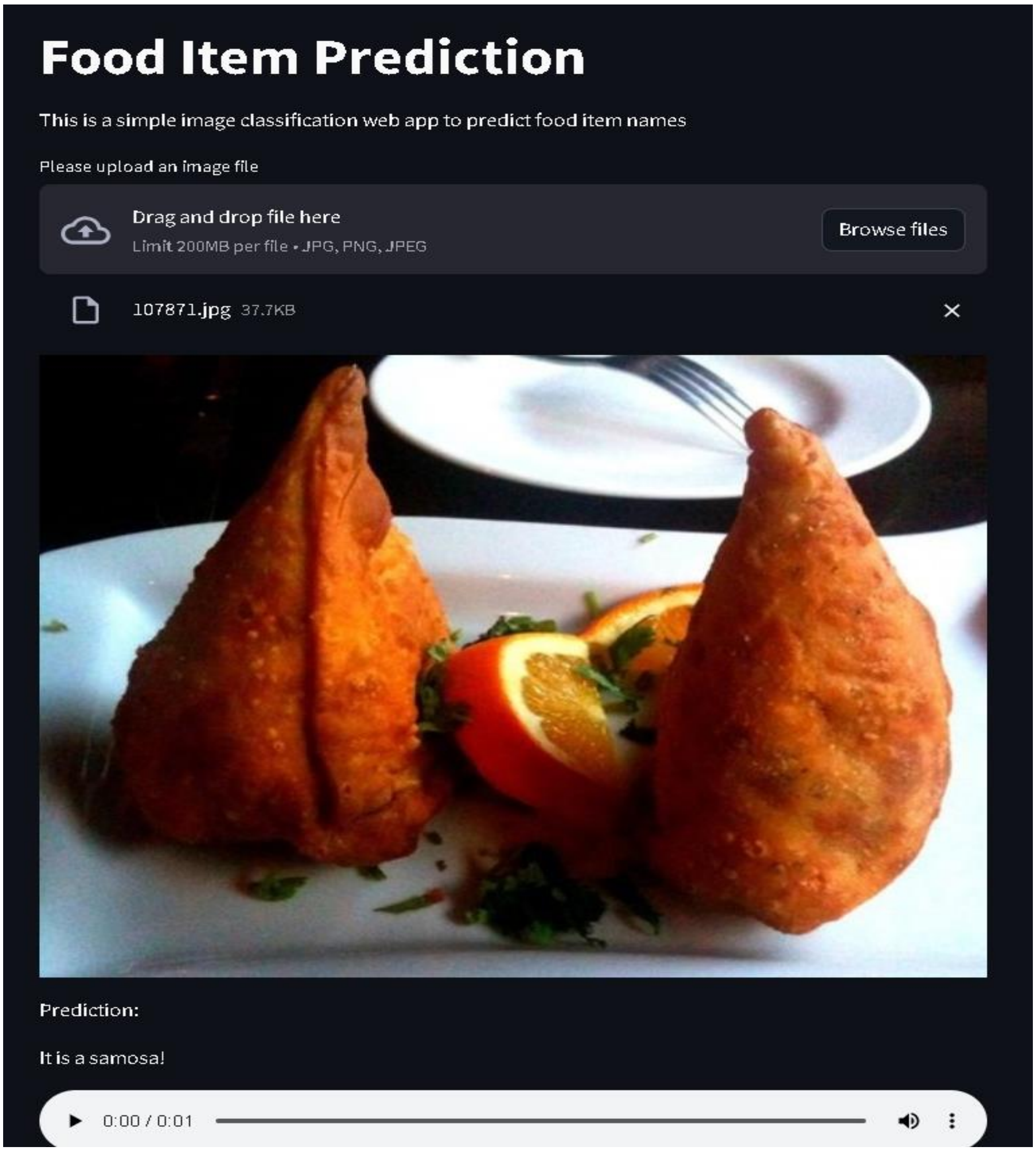
Please upload an image file

You haven't uploaded an image file

Figure 3.6.1- User Interface of Food recognition.

The user interface of a food recognition system is an interactive platform that enables users to interact with the application. It typically includes the following elements:

1. **Image Upload:** Users can upload an image of the food they want to recognize. This is often done through a simple file upload feature.
2. **Prediction Display:** After the image is uploaded, the system processes it and provides a prediction, indicating the name or type of the food in the image.



Food Item Prediction

This is a simple image classification web app to predict food item names

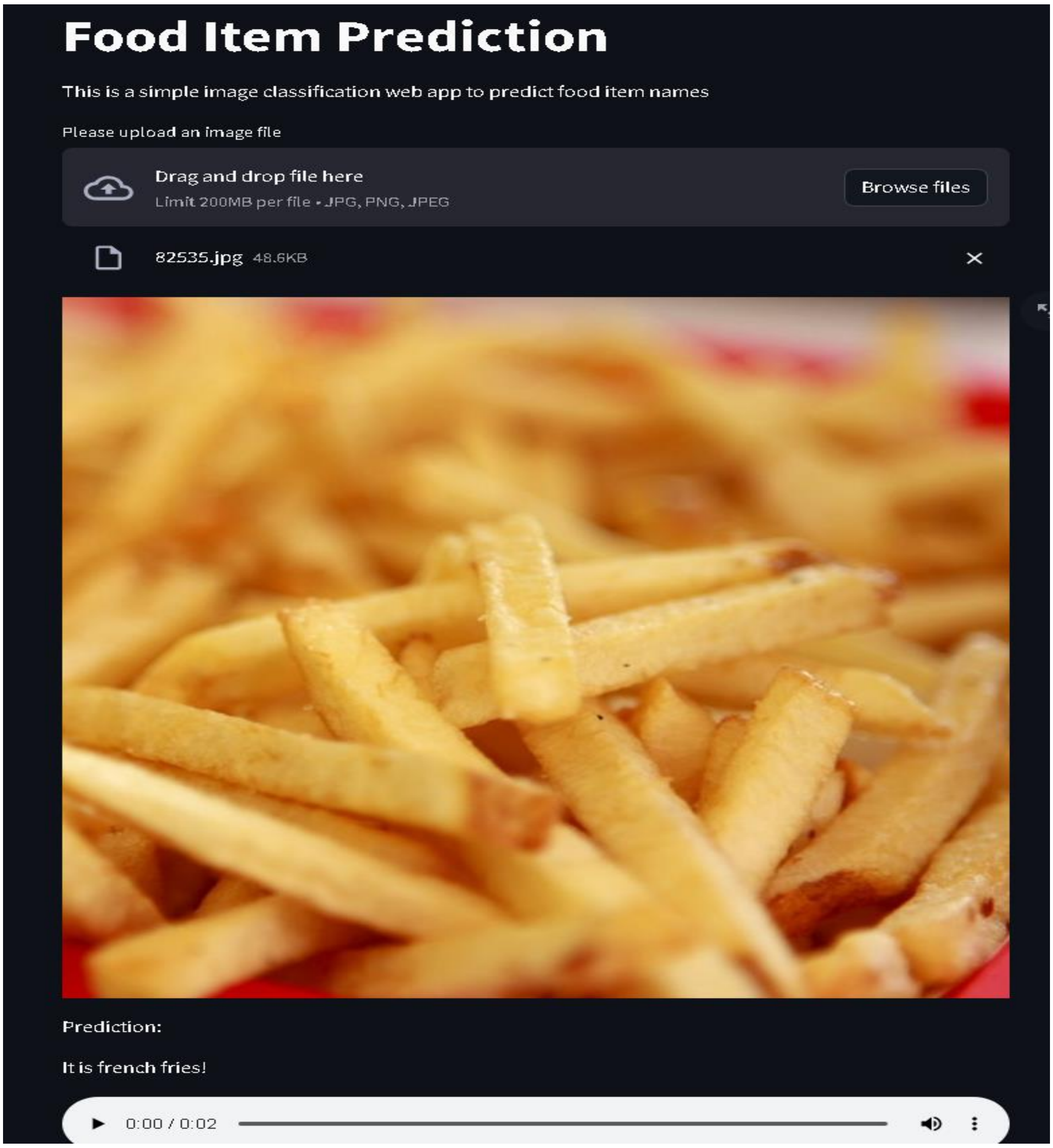
Please upload an image file

Prediction:

It is a samosa!

Figure 3.6.2- Model rightfully predicting the image as SAMOSA and the audio option will convert the text ‘samosa’ to speech

The model accurately predicts the image as "SAMOSA," and when the "samosa" prediction is displayed, the audio option converts the text "samosa" into clear and audible speech, enhancing the user experience by providing both visual and auditory feedback for accessibility.



Food Item Prediction

This is a simple image classification web app to predict food item names

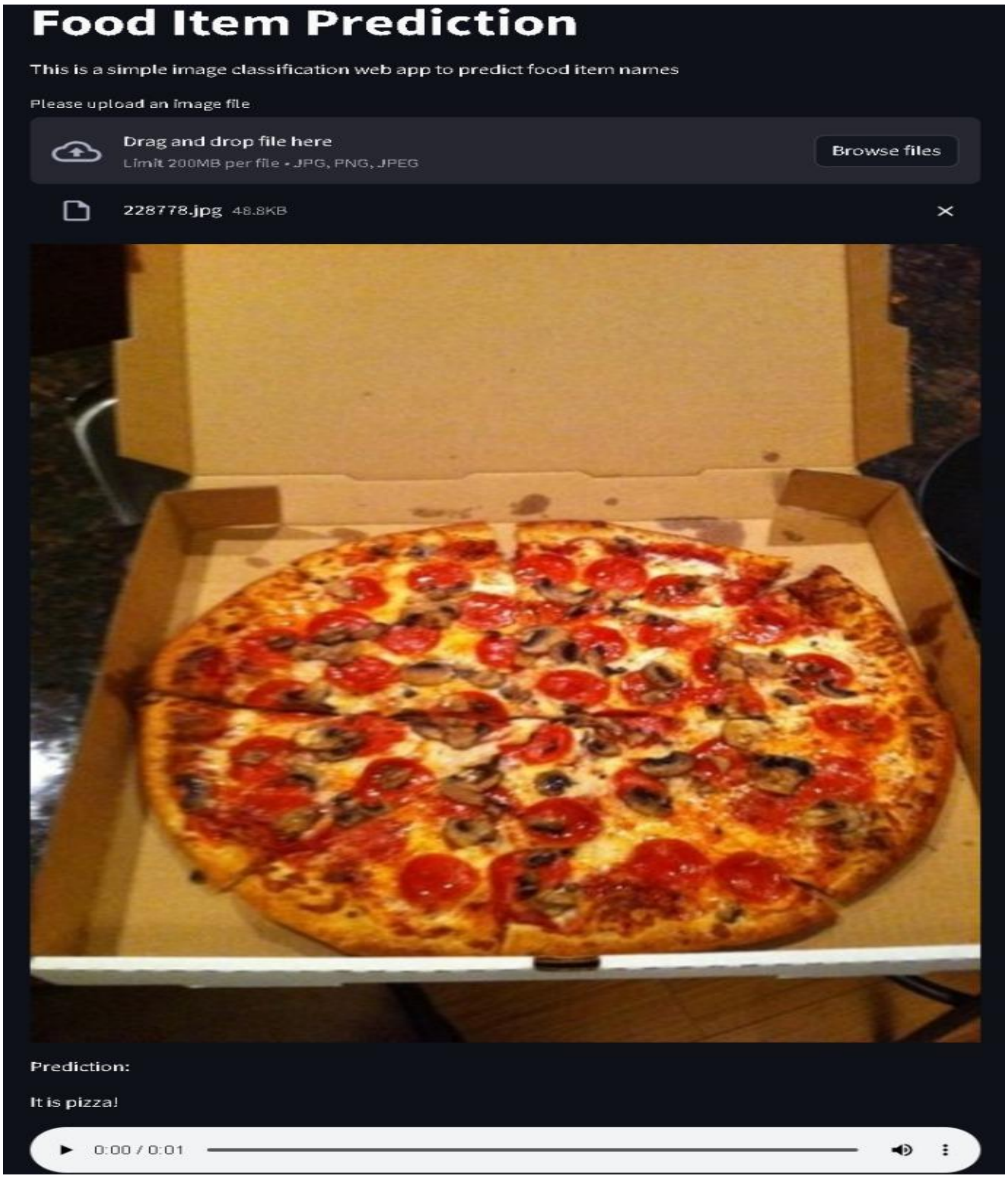
Please upload an image file

Prediction:

It is french fries!

Figure 3.6.3- Model rightfully predicting the image as FRENCH FRIES and the audio option will convert the text ‘French fries’ to speech

The model successfully identifies the image as "FRENCH FRIES." When the prediction is displayed as "French fries," the audio option swiftly converts this text into speech, ensuring a seamless and accessible user experience by audibly confirming the recognized food item, "French fries." This integration of visual and auditory feedback enhances the system's usability.



Food Item Prediction

This is a simple image classification web app to predict food item names

Please upload an image file

Prediction:

It is pizza!

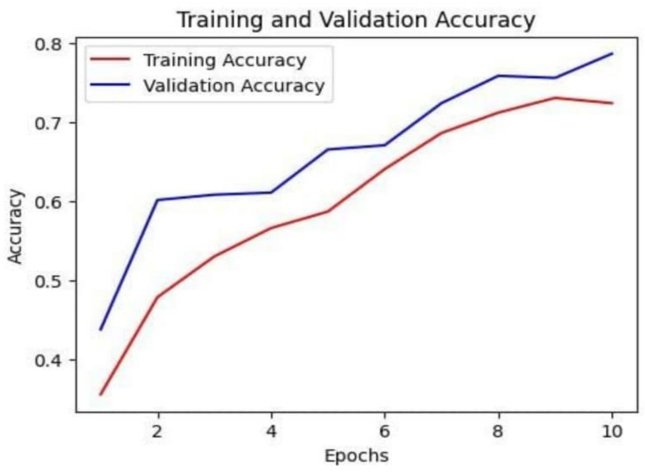
Figure 3.6.4- Model rightfully predicting the image as PIZZA and the audio option will convert the text ‘Pizza’ to speech.

The model accurately identifies the image as "PIZZA," and the audio feature promptly converts the text "Pizza" to speech, providing a seamless and accessible user experience by vocally confirming the recognized food item, "Pizza." This integration of visual and auditory feedback enhances the overall usability of the system

Chapter-4

Result Analysis

4.1 Comparison



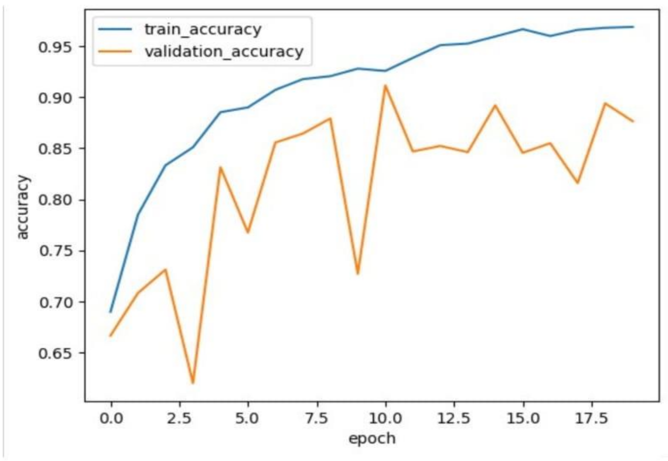


Figure 4.1.1 Comparison of Custom model (top) with the InceptionV3 pre-trained model (bottom)

formula:

Accuracy = (Number of Correct Predictions) / (Total Number of Predictions)

- At the end of each epoch, after the model has been trained on the training dataset, evaluate the model's performance on a validation dataset or a separate test dataset.
- Count the number of correct predictions on this dataset.
- Calculate the total number of predictions (the size of the dataset).
- Use the formula mentioned above to calculate the accuracy for that epoch.
- Record or log the accuracy value for each epoch to monitor the model's performance over time.

- The custom model has an accuracy of 79.04%, which means it correctly predicts the target class for approximately 79.04% of the examples in the evaluation dataset.
- The InceptionV3 model has a higher accuracy of 96.06%, indicating that it performs better on the same task, correctly classifying approximately 96.06% of the examples.

The goal of this figure 4.1.1 is to provide a visual comparison of how well the custom model and the Inception model perform over the course of training. This comparison can help you understand which model is better suited for your specific task. On the top, custom model and, on the bottom, The pre-trained Inception model, the validation accuracy is consistently smooth and exponential while that of the inception model has a lot of ups and downs (variations) and is not consistent.

4.2 Graph

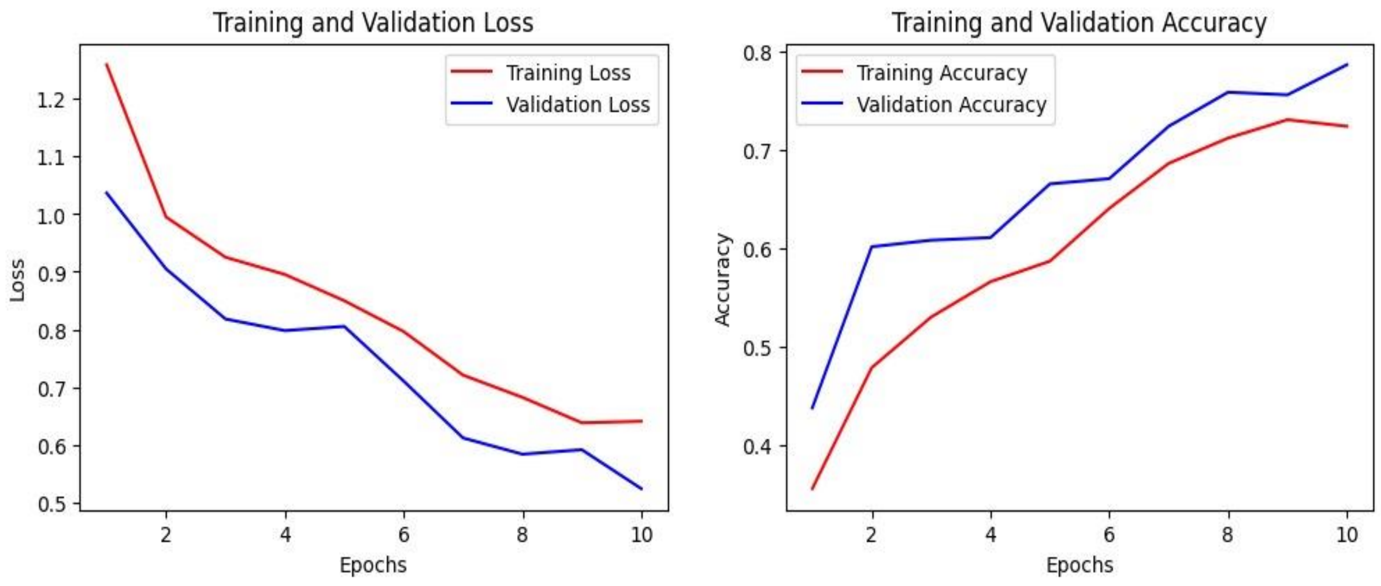


Figure 4.2.1 Custom model loss vs number of epochs (left)
Custom model accuracy vs number of epochs (right)

- Line Chart for Loss vs Epochs: This line chart shows the Training loss and Validation Loss of the custom model
- Line Chart for Accuracy vs Epochs: This line chart compares the training Accuracy and the Validation Accuracy.

Chapter-5

CONCLUSION

In our project, we aimed to create a food item classification system tailored to assist visually challenged individuals in recognizing diverse food types. We compared and employed two distinct approaches, incorporating a custom convolutional neural network (CNN) and the InceptionV3 pre-trained model. Our custom model was meticulously designed with multiple convolutional and pooling layers, as well as fully connected layers, enabling it to capture intricate features within food images. This model underwent training on a diverse dataset containing a wide range of food items.

In parallel, we harnessed the power of transfer learning with InceptionV3, a state-of-the-art pre-trained model. By fine-tuning this model for the food classification task, we built upon its existing knowledge and tailored it to specialize in recognizing various food items. A crucial aspect of our project was the integration of text-to-speech functionality, providing auditory feedback to visually challenged users. Upon recognizing a food item from an image, the system would verbally convey the identified item, enhancing accessibility for users who depend on auditory feedback. Our project successfully combines the capabilities of deep learning with accessible technology, offering visually challenged individuals the means to independently identify and savor a wide array of foods.

This project underscores the potential of pre-trained models for improved performance and usability. The inclusion of text-to-speech functionality enhances the overall accessibility of the system, making it a valuable tool for visually challenged individuals seeking food recognition autonomy.

References

[1] A. Sharma and G. Phonsa, "Image Classification Using CNN," Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2021, 27 Apr 2021.

[2] Mingyuan Xin & Yong Wang, "Research on image classification model based on deep convolution neural network," EURASIP Journal on Image and Video Processing, volume 2019, Article number: 40 (2019).

[3] Neha Sharma, Vibhor Jain, Anju Mishra, "An Analysis Of Convolutional Neural Networks For Image Classification," Procedia Computer Science, Volume 132, 2018, Pages 1571-1577. doi: 10.1016/j.procs.2018.05.198. This paper is available under a Creative Commons license.

[4] Lihua Luo, "Research on Image Classification Algorithm Based on Convolutional Neural Network," Journal of Physics: Conference Series, Volume 2083, 2. Computational Science, 2021, 032054. doi: 10.1088/1742-6596/2083/3/032054. This paper is published under license by IOP Publishing Ltd.

[5] Farhana Sultana, Abu Sufian, Paramartha Dutta, "Advancements in Image Classification using Convolutional Neural Network," Publisher: IEEE.

[6] Hitesh Kumar Sharma, et al., "Deep Learning based Binary Classification of Paintings and Photographs using CNN Model," University of Petroleum & Energy Studies, India.

[7] Weblinkshttps://en.wikipedia.org/wiki/Computer_vision#Recognition

- https://www.youtube.com/watch?v=IIs1flEtpQk
- https://www.geeksforgeeks.org/data-visualization-using-matplotlib/
- https://pytorch.org/hub/research-models/