ABSTRACT

We have over a decillion diverse cuisine which furcate to create hundreds of niche styles of diversified flavors, tastes, and appearance. People appreciate food and enjoy food photography. So, they post food images on social media platforms like Instagram. Behind each flavorful bite, there is an unspoken narrative illustrated in a recipe. Unfortunately, just through a glance at the food image, we are unable to understand the approach to its preparation procedure. In this project, we have introduced a Reverse Cooking system that revives the cooking recipes in the form of recipe names, ingredients, and cooking procedures from the inputted food image. The entire model is evaluated on a large-scale Recipe dataset and generates highly accurate predictions by leveraging the food image. A key challenge may pose during this model building as dishes of distinct cuisines may look similar regarding their cooking procedures. To solve this, the model has been trained using Convolutional Neural Networks to categorize the food images into various categories and output a matched recipe. Combining it with a high-level classifier improves retrieval performance. To summarize, the model works on automated image recognition over a photographed dish and the output of the matched recipe subsequently.

LIST OF FIGURES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1.INTRODUCTION

1.INTRODUCTION

1.1 PROJECT SCOPE

This project is titled as “Image to recipe prediction system”. This software provides facility to upload the dataset and get to know about the various food recipes from the images. This project uses machine learning methodologies to predict the food recipes based on given dataset. We use convolutional neural networks to predict the food recipe from the dataset given.

1.2 PROJECT PURPOSE

This has been developed to facilitate the extracting the features of images from the uploaded dataset and to predict the respective food recipe from the features obtained from the dataset. The images from dataset uploaded are classified and summarized by a feature-based image classification technique. This information will be more helpful to the user to analyze the food recipes.

1.3 PROJECT FEATURES

The main feature of this project is to perform the image classification on the given images using CNN algorithm. The CNN is a deep learning algorithm that is employed for image classification. It uses five different filter sizes to each image and GlobalMaxPolling1D layers are applied to each consecutive layer. The outputs are all connected to the final dense layer. The output of this project is to classify the given image into a respective food recipe by the CNN model.

2.SYSTEM ANALYSIS

2.SYSTEM ANALYSIS

SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, "what must be done to solve the problem?" The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to be done.

2.1 PROBLEM DEFINITION

Motivated by different real-world applications, researchers have considered a wide range of problems over a variety of different types of corpora. We now examine the key concepts involved in these problems. This discussion also serves as a loose grouping of the major problems, where each group consists of problems that are suitable for similar treatment as learning tasks. One set of problems share the following general character: a food image , where in it is assumed that as a unknown food image , classify the image into respective food recipe and also predict food ingredients present int the food recipe.

2.2 EXISTING SYSTEM

In the existing system for reverse cooking , methods and algorithms are not that much accurate in classification. In general in existing systems classification are ANN due to accuracy dependency of quality of classifications.

2.2.1 LIMITATIONS OF EXISTING SYSTEM

- More classification.
- Time consuming.

The main disadvantages of using ANN is significantly slower due to an operation such as Maxpool. As ANN has several layers then the training process takes a lot of time in image processing, if the computer doesn't consist of a good GPU.

2.3 PROPOSED SYSTEM

The aim of proposed system is to develop a system of improved facilities. The image of the food is collected from the web and is given to the model. Image of the food undergoes pre-processing and modelled to extract fine-tuned features and thereby, predict the Ingredients and the cooking instructions of the inputted image. This process is known as reverse-cooking

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

- The Convolutional Neural Networks (CNN or ConvNet) are complex feed forward neural networks. CNNs are used for image classification and recognition because of its high accuracy.
- The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision.
- CNN is also computationally efficient.

2.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company.

Three key considerations involved in the feasibility analysis

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

2.3.1 ECONOMIC FEASIBILITY

Development of this application is highly economically feasible. The organization needed not spend much money for the development of the system already available. The only thing is to be done is making an environment for the development with an effective supervision. If we are doing so, we can attain the maximum usability of the corresponding resources. Even after the development, the organization will not be in condition to invest more in the organization. Therefore, the system is economically feasible.

2.3.2 TECHNICAL FEASIBILITY

We can strongly say that it is technically feasible, since there will not be much difficulty in getting required resources for the development and maintaining the system as well. All the resources needed for the development of the software as well as the maintenance of the same is available in the organization here we are utilizing the resources which are available already.

2.3.3 BEHAVIORAL FEASIBILITY

Whatever we think need not be feasible. It is wise to think about the feasibility of any problem we undertake. Feasibility is the study of impact, which happens in the organization by the development of a system. The impact can be either positive or negative. When the positives nominate the negatives, then the system is considered feasible. Here the feasibility study can be performed in two ways such as technical feasibility and Economical Feasibility.

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- Processor : Intel Dual Core@ CPU 2.90GHz.
- Hard disk : 16GB and Above.
- RAM : 4GB and Above.

2.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements.

- Operating system : Windows 8,10
- Languages : Python
- Backend : Machine Learning
- IDE : PyCharm

3.ARCHITECTURE

3.ARCHITECTURE

3.1 PROJECT ARCHITECTURE

This project architecture shows the procedure followed for image classification of given data using machine learning, starting from input to final prediction.

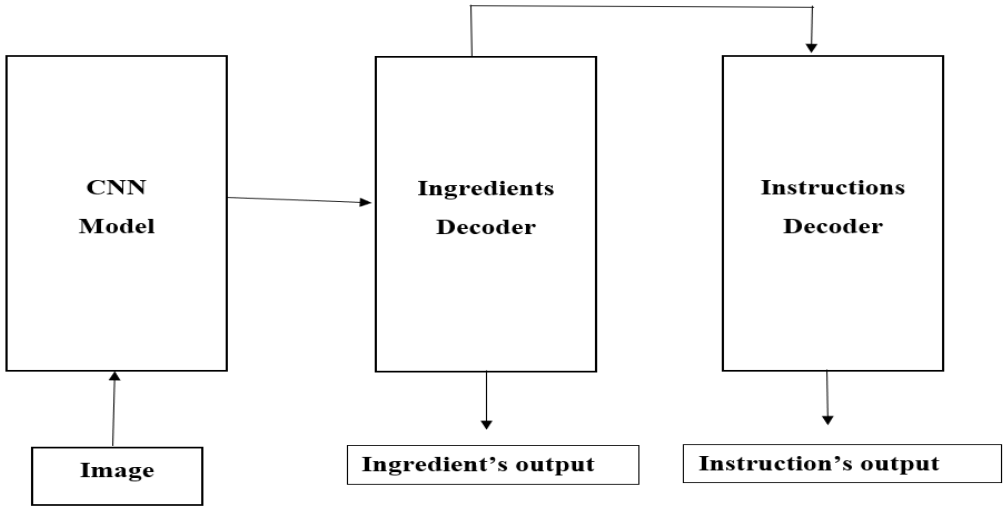


Figure 3.1: Project Architecture of Opinion Mining on Feedback Management System

3.2 DESCRIPTION

The project has been classified into six modules (or stages) in a sequential order that the Input image has to go through to remodel itself into a valuable Net numeric Score that proves to be an asset in commercial business. This modular approach of the project is shown below sequentially.

Module 1: DATA CLEANING

Besides unraveling the hidden patterns and insights, Data Exploration allows one to take up initial steps in building a distinctly accurate model.

- Major time needs to be spent on data exploration, cleaning, and preparation as this would eat away a crucial portion of project time.
- It also supports superior and well-curated analysis and improved business intelligence for processing and decision making.

Although our core_data_recpie.csv dataset includes 1125 Multi-cuisine recipes, but still a lot of them cannot be used due to lack of appropriate quality. Many community cooking blogs contain multiple recipes of a single food dish that are largely unstructured. As part of the Data Cleaning pipeline, we have identified few images and textual information and cleaned them as follow:

Instructions: Manual Investigation of several records in the given dataset, proved some users have given URLs or emoticons and other special symbols in the cooking instructions text. The distribution of Instruction length was observed to be Right skewed (i.e., Instruction length in terms of the number of characters that fall in the positive region).

Ingredients: The ingredient list makes up most of the unstructured portion of the dataset. It consists of roughly around 10,000 unique Ingredients mainly because they also contain pronouns and adjectives (e.g., Turkey Black Bean Burgers instead of only Bean Burgers).

- **Removal or replacement of Special symbols** (such as @, ! , -, *,) with blank spaces.
- **Handling Compound words:** When two or more words are concatenated to form a new word that has a completely different definition. Such words are called Compound words.

Image Scrapping: Image scraper used certain requests library to extract food images from the specified websites. The request libraries involve Beautiful Soup and pandas to export scraped data (i.e. image URLs) and present output data into our core_data_recipe.csv file. We assigned an appropriate web driver to pick the URL from which we scraped image links and created a list data structure for storing.

Image Resolution: For training an accurate model, at least four food images of the same dish with adequate resolutions are necessary. Recipes without corresponding images were also retained as they can be matched with the ones with images. We have resized the food images to 63*63*3 pixels for our model input.

Module 2: DATA PRE-PROCESSING :

Importing Libraries: We begin with importing the libraries that would be required to perform certain tasks in the code. Library is basically a set of built-in modules by the developer during the installation which can be called and used whenever required within the program.

```
import numpy as np
```

```
import pandas as pd
```

Splitting the Dataset: The entire dataset is split into Training Set, and Validation Set, and Test Set of similar distribution. A 60-20-20 split was performed on the existing dataset into three categories as follows:

- **Training Set:** This contains 60% of the entire dataset, which is required used to train the CNN model.
- **Test Set:** This contains 20% of the entire dataset, which is used in estimating the performance of the model in terms of accuracy and loss.
- **Validation Set:** This contains 20% of the entire data, which is used to fine-tune the hyperparameters to determine the model learning rate.

Split count of Food Images:

- Count of Training images: 1000
- Count of Test images: 125
- Count of Validation images: 20

Input pipeline: An Input pipeline that is nothing but the core_data_recipe dataset, loads the training examples in the form of a **tuple (n, img x(ing), x(inst))** to the model.

- Here n is the unique recpie_number.
- img represents the food images in the dataset.
- x(ing) denotes the Ingredients list and;
- x(inst) has the cooking instructions along with timings in the form of text.

Images: The cleaned dataset includes 4 images of each food item. During the training phase, an image is selected randomly from the remaining 3 images for better learning. A lesser training set leads to Overfitting (i.e.: For instance, the model recognizes food images in the training data. So, the accuracy of the training set will be greater than that of the Test set which is termed as overfitting).

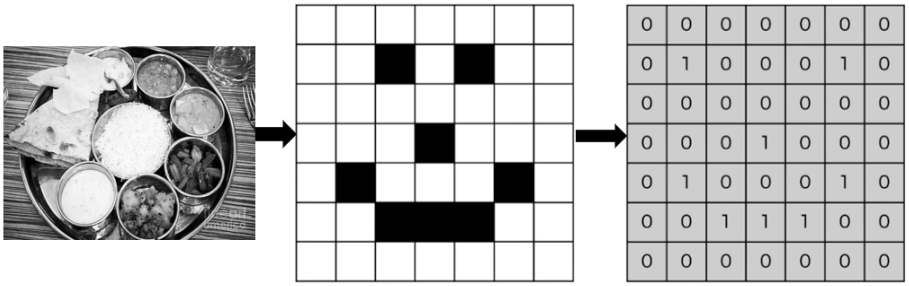
Ingredients: Ingredients are tokenized by using a word encoder with fixed-sized vocabulary. Breaking a character string into pieces, called Tokens, and throwing away certain characters at the same time, including punctuation and special characters. A special delimiter '^' is used to separate ingredients of the ingredients list in the data set. The ingredients in the ingredients are placed in a random order as the order does not affect the model learning.

Title and Cooking Instructions: The recipe title is generated using the major ingredients in the ingredients list. We have concatenated the title and the cooking instructions. Cooking instructions also include the time required to perform every instruction and the total time becomes the sum of such time slots.

Module 3: BUILDING CNN MODEL

A renowned Deep learning class that is employed Image Recognition is CNN. **Inspiration of VGGNet:** The most preferred VGGNet was designed by Visual Geometry Group and stood as the runner-up in the ImageNet competition as the most effective Feature extraction model. Inspired from the VGG-16 model which is 16 hidden layers and pre-trained on more than one million images of around 1000 objects from the ImageNet database, we have built a model of the same architecture for ingredient recognition for the food image.

Implementation CNN Architecture in Reverse Cooking: Every image consists of pixels ranging from 0 to 255 and three color channels namely- Red, Green, and Blue (RGB color pallet). So, to simply we made this black and white image with 0 and 1 pixels. Suppose we are given food image of Indian Thali as shown in Fig 3.2:



Step 1: Convolution- Here we have taken an image of 63*63*30 pixels and for feature detection, we make use of a filter of size 4*4*4. The filter is rolled throughout the image and a cartesian product is performed to generate a feature map. This is repeated until the entire image is featured in the feature map by moving the filter horizontally and vertically. After creating feature maps, we arrange them together to form our first convolutional layer.

Step 2: Max Pooling - Upon obtaining the Convolution layer, each feature is taken and Max pooling is done. Max pooling is a pooling function that chooses the maximum element from the feature map covered by the filter of size 4*4*4. After performing max pooling on all the features map, we obtain a matrix with all the maximum elements. Such a matrix is called a Pooled Feature Map which highlights the most interesting and present feature in the food image. Pile of Pooled Feature Map makes up the Pooling Layer. This layer has only the most significant features of the food image which paves way for quicker computations.

Step 3: Flattening- Conversion of the obtained Pooled Feature map data into a 1-D array for inputting it to the next subsequent layer for faster processing is called Flattening. A flattened layer is nothing but an exclusive long feature vector vertically. The flattened layer is connected to the CNN model, which is the fully connected layer. The output of the flattened layer is used by the Dense layer of the subsequent neural network model.

Step 4: Full Connection- Fully connected layers are inputted with Pooled features map obtained during Pooling, which undergoes flattening before being inputted to a fully connected layer. CNN Model consists of three distinct layers namely:

- **Input layer** which comprises of Input data.
- **Hidden layers** which include activation nodes called neurons; and
- An **Output layer** includes one or more neurons, whose outputs are the final output of the network.

Unlike the Input layer and hidden layers which use the ReLU activation function, the *S*oftMax activation function is employed in the output layer. SoftMax to obtain class probabilities of the input for classification. Whereas ReLU is a max function performed on the neurons in the layer to give only positive and zero values.

Translation of features into another form in Sequence transduction models is quite challenging since the input and output are both variable-length sequences. Such types of models are handled with the help of Encoder and Decoder sequential architecture. The Reverse cooking model also has 1 encoder- Image encoder and 2 decoders namely: Ingredient decoder and cooking Instruction decoder.

Module 4: IMAGE ENCODER

Generally, Encoder takes a variable-length input sequence and then transforms it into fixed-size output. Here, the image encoder maps an input image img into a feature representation X which consists of the number of image features and the embedding dimension. Upon image data encoded to an integer, the embedding layer represents each word in the form of a unique numeric value. Typically, the decoder maps the encoded output of a fixed length to a variable-length sequentially by taking inputs and generating outputs at each intervals of time.

Module 5: INGREDIENTS DECODER

It is a transformer decoder network that is conditioned on the image features x to produce feature vectors denoting the number of ingredients. The decoder output of the will be inputted to a linear output layer followed by a SoftMax activation function to generate the predictions. SoftMax activation ensures that the output probabilities sum up to 1.

Module 6: COOKING INSTRUCTIONS DECODER

The output of the Ingredients decoder is given to the cooking instruction decoder for the generation of food recipes. It is conditioned on both the embedded image img and the ingredients features x obtained from the Ingredients decoder. Upon processing, the output of the Instruction decoder was fed into a linear layer followed by the SoftMax activation function to generate probabilities over the vocabulary of cooking instructions text.

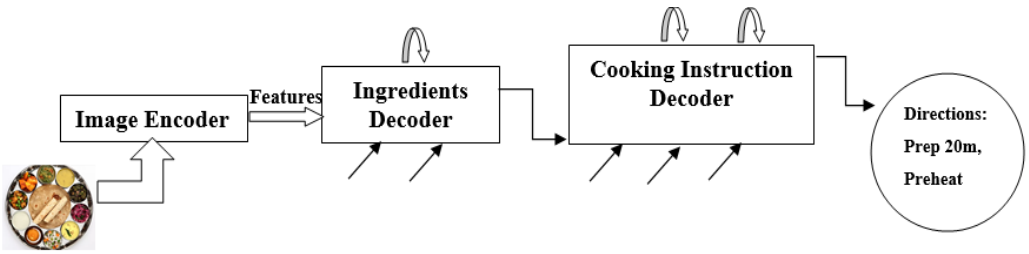


Figure 3.2 : Flow control of Encoder and Decoders in Reverse Cooking

3.3 USE CASE DIAGRAM

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors.

An actor is represents a user or another system that will interact with the system you are modeling. A use case is an external view of the system that represents some action the user might perform in order to complete a task.

Contents:

- Use cases
- Actors
- Dependency, Generalization, and association relationships
- System boundary

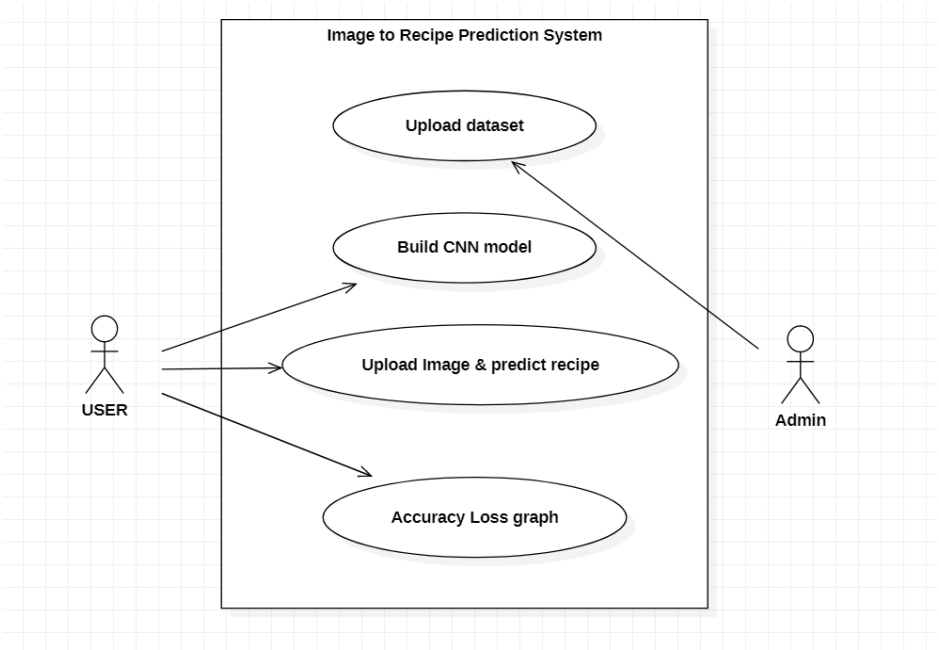


Figure 3.3 : Use case Diagram for Image to Food Recipe Prediction System

3.4 CLASS DIAGRAM

Class diagrams model social organization and its contents using design elements like classes, packages, and objects, thereby describing various objects used during a system and their relationships." They define the cognitive, requirement, and functionality paradigms when developing a system by illustrating the classes in the program, attributes, functions of each class, and the relationship that exists between each class.

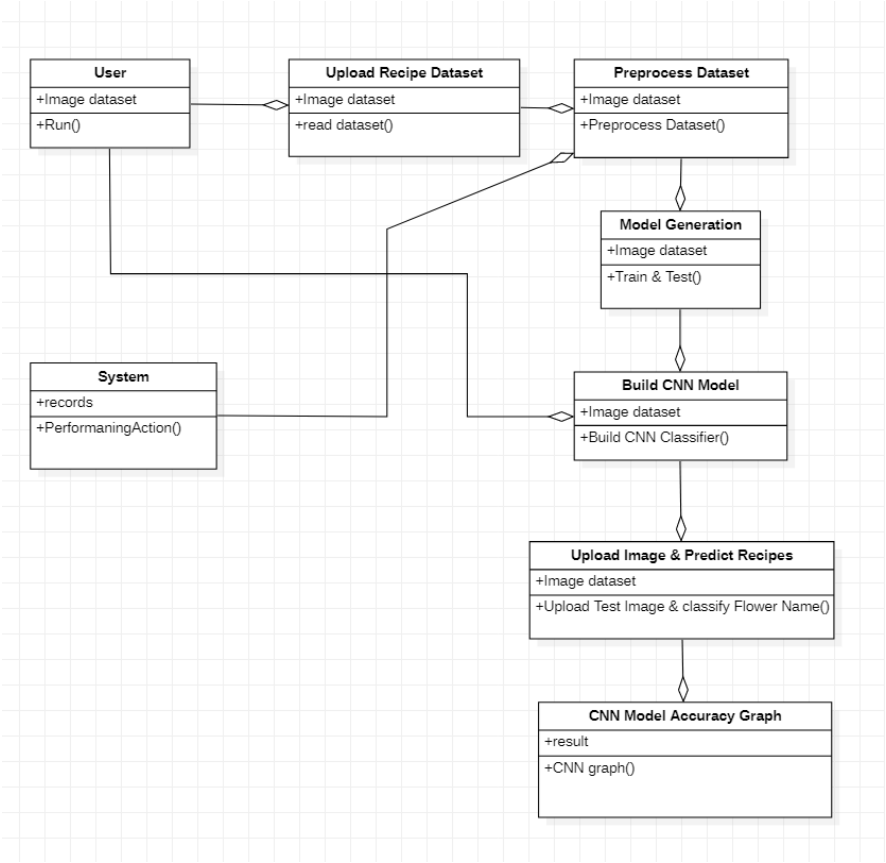


Figure 3.4: Class Diagram for Image to Food Recipe Prediction System

3.5 SEQUENCE DIAGRAM

Sequence diagrams in UML shows how object interact with each other and the order those interactions occur. It's important to note that they show the interactions for a particular scenario. The processes are represented vertically and interactions are show as arrows. This article explains the purpose and the basics of Sequence diagrams.

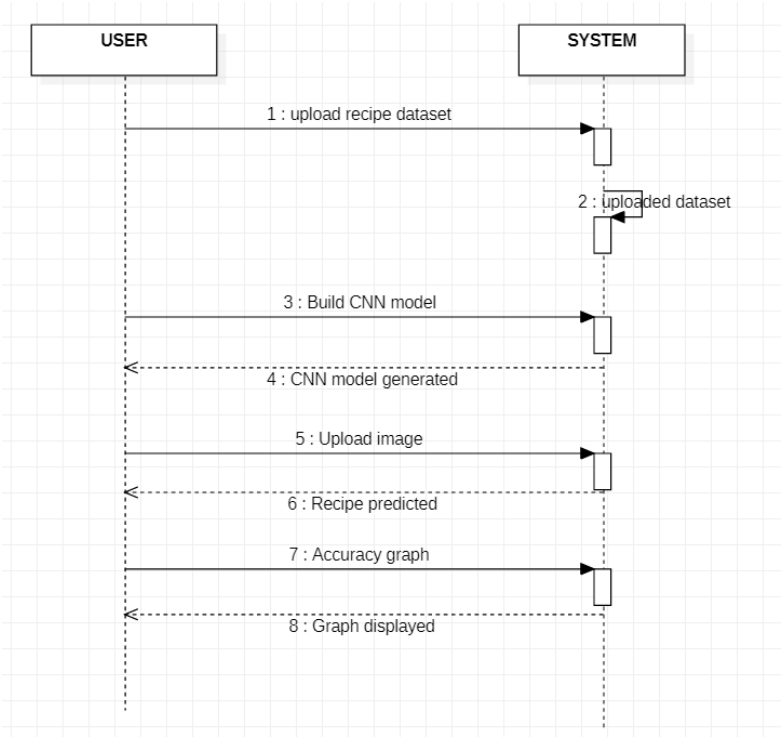


Figure 3.5: Sequence Diagram for Image to Food Recipe Prediction System

3.6 ACTIVITY DIAGRAM

It describes about flow of activity states.

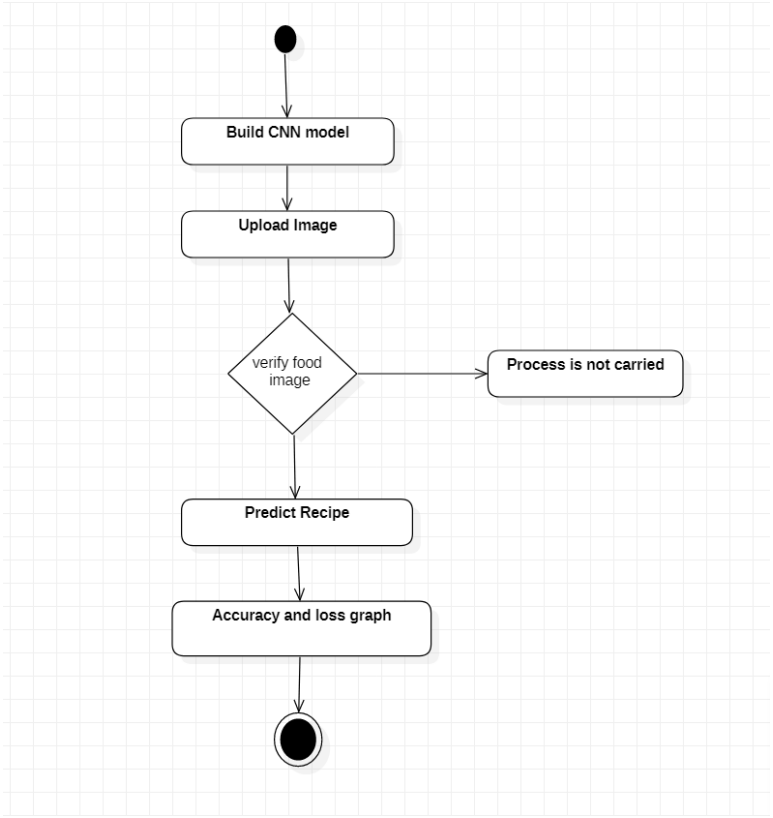


Figure 3.6: Activity Diagram for Image to Food Recipe Predcition System

4.IMPLEMENTATION

4. IMPLEMENTATION

4.1 SAMPLE CODE

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
import matplotlib.pyplot as plt
from tkinter.filedialog import askopenfilename
import numpy as np
import pandas as pd
from Recipe import *
import os
import cv2
from keras.utils.np_utils import to_categorical
from keras.layers import  MaxPooling2D
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D
from keras.models import Sequential
from keras.models import model_from_json
import pickle
import ast

main = tkinter.Tk()
main.title("Inverse Cooking: Recipe Generation from Food Images")
main.geometry("1300x1200")

global filename
global classifier
recipe_list = []
global dataset
```

```
def uploadDataset():
    textarea.delete('1.0', END)
    global filename
    global dataset
    recipe_list.clear()
    filename = filedialog.askopenfilename(initialdir="Dataset")
    pathlabel.config(text=filename)
    textarea.insert(END,'Dataset loaded\n\n')

    dataset = pd.read_csv(filename,nrows=1000)
    for i in range(len(dataset)):
        r_id = dataset.get_value(i, 'recipe_id')
        r_name = dataset.get_value(i, 'recipe_name')
        ingredients = dataset.get_value(i, 'ingredients')
        nutritions = dataset.get_value(i, 'nutritions')
        cooking = ast.literal_eval(dataset.get_value(i, 'cooking_directions')).get('directions')
        r_name = r_name.strip().lower()
        obj = Recipe()
        obj.setRecipeID(r_id)
        obj.setName(r_name)
        obj.setIngredients(ingredients)
        obj.setNutritions(nutritions)
        obj.setCooking(cooking)
        recipe_list.append(obj)
    indian = np.load('index.txt.npy',allow_pickle=True)
    for i in range(len(indian)):
        recipe_list.append(indian[i])
    obj = recipe_list[len(recipe_list)-1]
    print(obj.getName())
    textarea.insert(END,"Recipes data loaded\n")

def buildCNNModel():
    textarea.delete('1.0', END)
    global classifier
```

```
    if os.path.exists('model/model.json'):
        with open('model/model.json', "r") as json_file:
            loaded_model_json = json_file.read()
            classifier = model_from_json(loaded_model_json)
        classifier.load_weights("model/model_weights.h5")
        classifier._make_predict_function()
        print(classifier.summary())
        f = open('model/history.pckl', 'rb')
        data = pickle.load(f)
        f.close()
        acc = data['accuracy']
        accuracy = acc[1] * 100
        textarea.insert(END,"CNN training process completed with Accuracy = "+str(accuracy))
    else:
        encoding_dim = 32
        X = X_train.reshape(X_train.shape[0],(64 * 64 * 3))
        print(X.shape)
        input_img = keras.Input(shape=(X.shape[1],))
        encoded = layers.Dense(encoding_dim, activation='relu')(input_img)
        decoded = layers.Dense(Y_train.shape[1], activation='softmax')(encoded)
        autoencoder = keras.Model(input_img, decoded)
        encoder = keras.Model(input_img, encoded)
        encoded_input = keras.Input(shape=(encoding_dim,))
        decoder_layer = autoencoder.layers[-1]
        decoder = keras.Model(encoded_input, decoder_layer(encoded_input))
        autoencoder.compile(optimizer='adam', loss='categorical_crossentropy', metrics =
['accuracy'])
        hist = autoencoder.fit(X, Y_train, batch_size=16, epochs=10, shuffle=True, verbose=2)

def predict():
    textarea.delete('1.0', END)
    filename = filedialog.askopenfilename(initialdir="testImages")
    image = cv2.imread(filename)
    img = cv2.resize(image, (64,64))
```

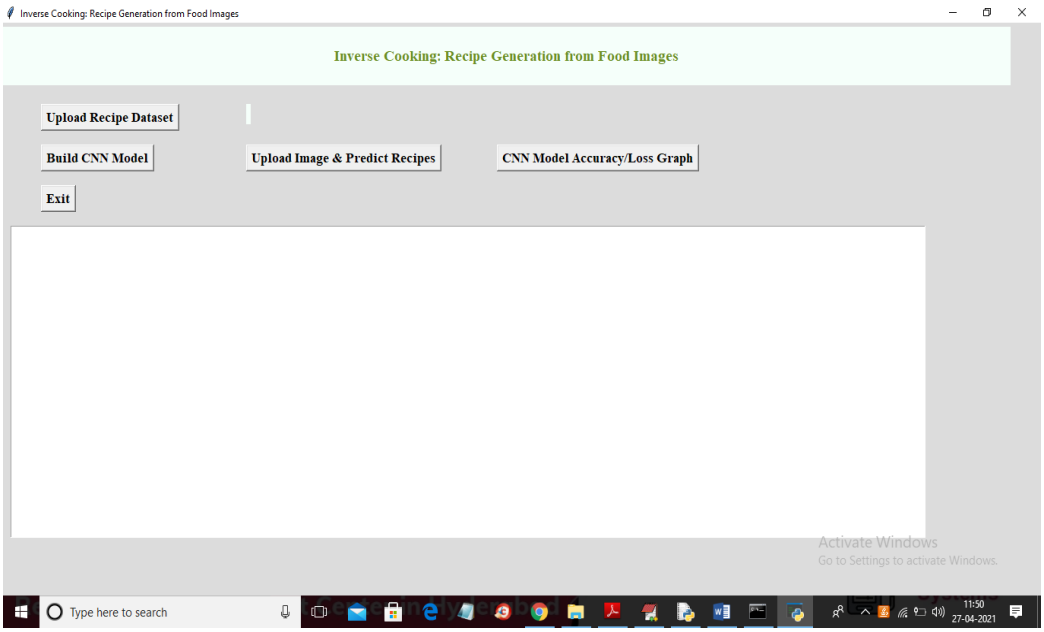
```
    im2arr = np.array(img)
    im2arr = im2arr.reshape(1,64,64,3)
    img = np.asarray(im2arr)
    img = img.astype('float32')
    img = img/255
    preds = classifier.predict(img)
    predict = np.argmax(preds)
    if predict > 0:
        predict = predict - 1
    print(predict)
    obj = recipe_list[predict]
    textarea.insert(END,"Recipe Name\n")
    textarea.insert(END,obj.getName()+"\n\n")
    textarea.insert(END,"Ingredients Details\n")
    textarea.insert(END,obj.getIngredients()+"\n\n")
    textarea.insert(END,"Cooking Details\n")
    textarea.insert(END,obj.getCooking()+"\n\n")
    textarea.insert(END,"Nutritions Details\n")
    textarea.insert(END,obj.getNutritions()+"\n\n")

    img = cv2.imread(filename)
    img = cv2.resize(img, (800,400))
    cv2.putText(img, 'Receipe Name : '+obj.getName(), (10, 25),
cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 255, 255), 2)
    cv2.imshow('Receipe Name : '+obj.getName(), img)
    cv2.waitKey(0)

def graph():
    f = open('model/history.pckl', 'rb')
    data = pickle.load(f)
    f.close()
    accuracy = data['accuracy']
    loss = data['loss']

    plt.figure(figsize=(10,6))
```

```
    plt.grid(True)
    plt.xlabel('Iterations')
    plt.ylabel('Accuracy/Loss')
    #plt.plot(loss, 'ro-', color = 'red')
    #plt.plot(accuracy, 'ro-', color = 'green')
    plt.plot(loss, 'ro-', color = 'blue')
    plt.plot(accuracy, 'ro-', color = 'orange')
    plt.legend(['Loss', 'Accuracy'], loc='upper left')
    #plt.xticks(wordloss.index)
    plt.title('Recipe CNN Accuracy & Loss Graph')
    plt.show()

def close():
    main.destroy()

font = ('times', 14, 'bold')
title = Label(main, text='Inverse Cooking: Recipe Generation from Food Images')
title.config(bg='mint cream', fg='olive drab')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 13, 'bold')
uploadButton = Button(main, text="Upload Recipe Dataset", command=uploadDataset)
uploadButton.place(x=50,y=100)
uploadButton.config(font=font1)

pathlabel = Label(main)
pathlabel.config(bg='mint cream', fg='olive drab')
pathlabel.config(font=font1)
pathlabel.place(x=320,y=100)

cnnButton = Button(main, text="Build CNN Model", command=buildCNNModel)
cnnButton.place(x=50,y=150)
```

```
cnnButton.config(font=font1)

predictButton = Button(main, text="Upload Image & Predict Recipes", command=predict)
predictButton.place(x=320,y=150)
predictButton.config(font=font1)

graphButton = Button(main, text="CNN Model Accuracy/Loss Graph", command=graph)
graphButton.place(x=650,y=150)
graphButton.config(font=font1)

closeButton = Button(main, text="Exit", command=close)
closeButton.place(x=50,y=200)
closeButton.config(font=font1)

font1 = ('times', 12, 'bold')
textarea=Text(main,height=20,width=150)
scroll=Scrollbar(textarea)
textarea.configure(yscrollcommand=scroll.set)
textarea.place(x=10,y=250)
textarea.config(font=font1)

main.config(bg='gainsboro')
main.mainloop()
```

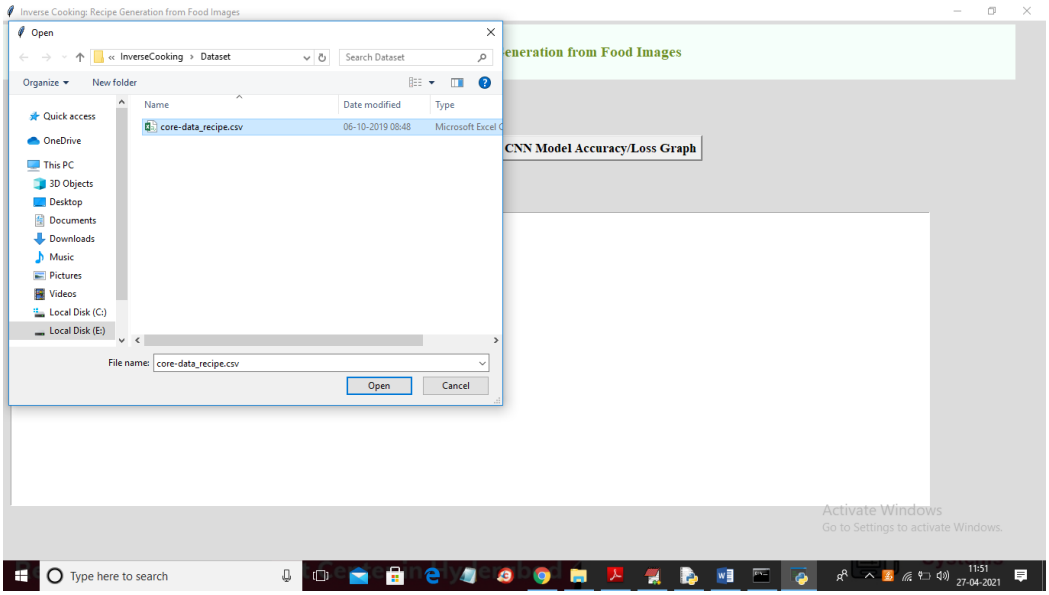
5.SCREENSHOTS

To run project double click on ‘run.bat’ file to get below screen



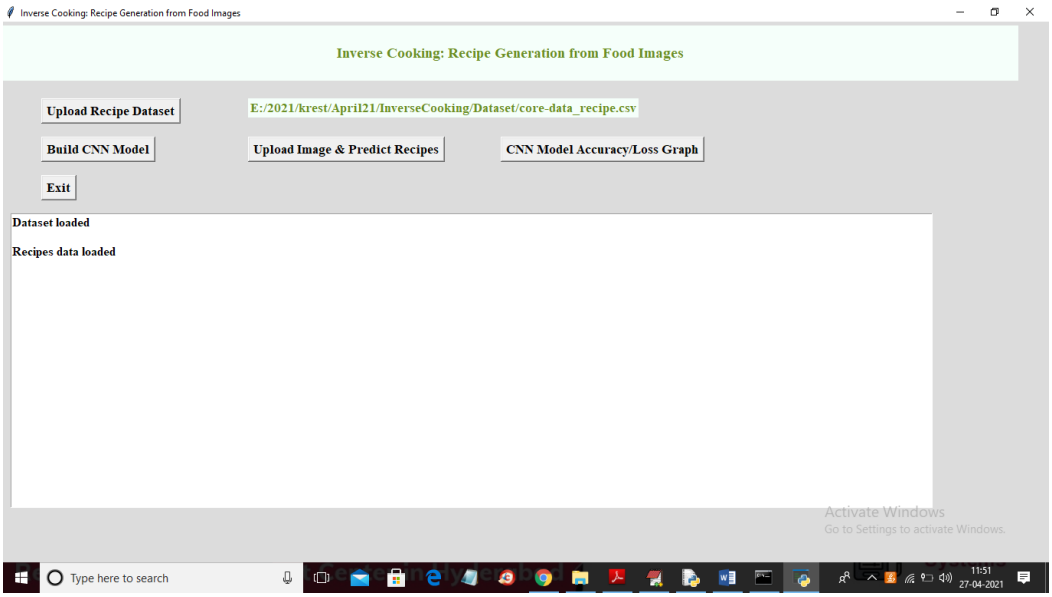
Screenshot 5.1 Application GUI

In above screen click on ‘Upload Recipe Dataset’ button to upload dataset



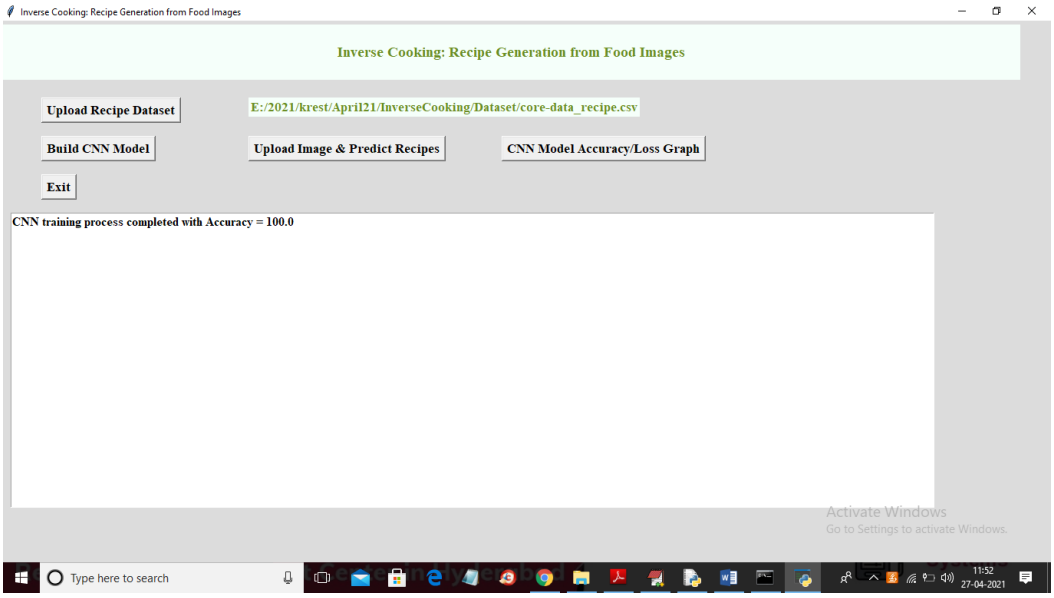
Screenshot 5.2 Upload Dataset

In above screen selecting and uploading recipe dataset and then click on ‘Open’ button to load dataset and to get below screen



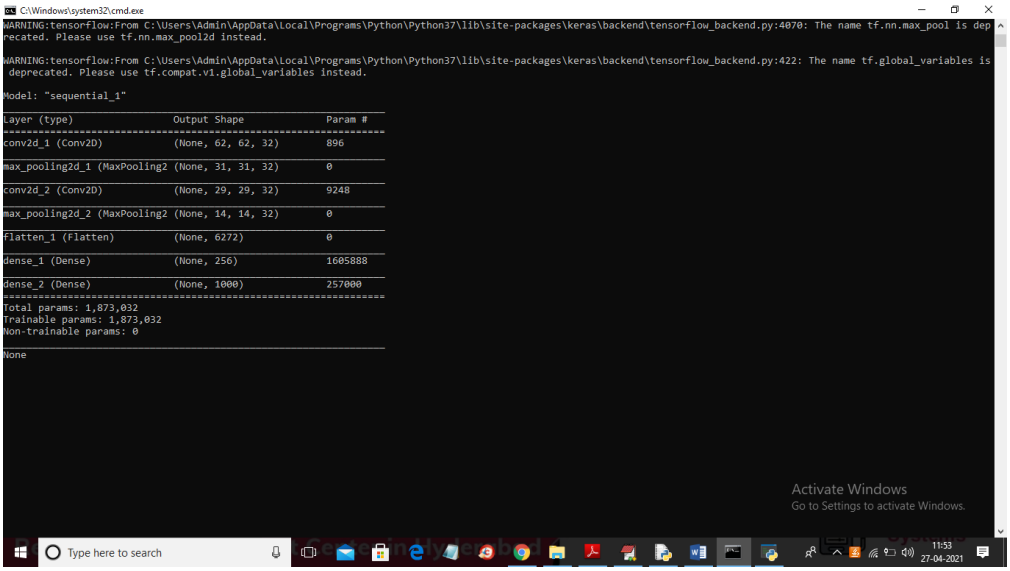
Screenshot 5.3 Dataset Uploaded

In above screen dataset loaded and now click on 'Build CNN Model' button to build CNN on above dataset



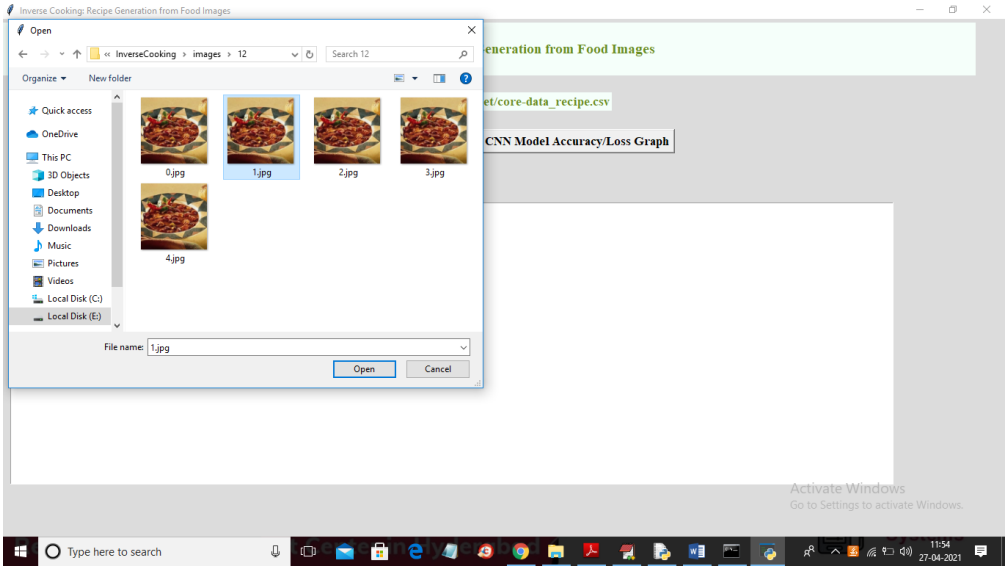
Screenshot 5.4 CNN Build

In above screen CNN model generated and we got prediction accuracy as 100% and in below screen you can see CNN details



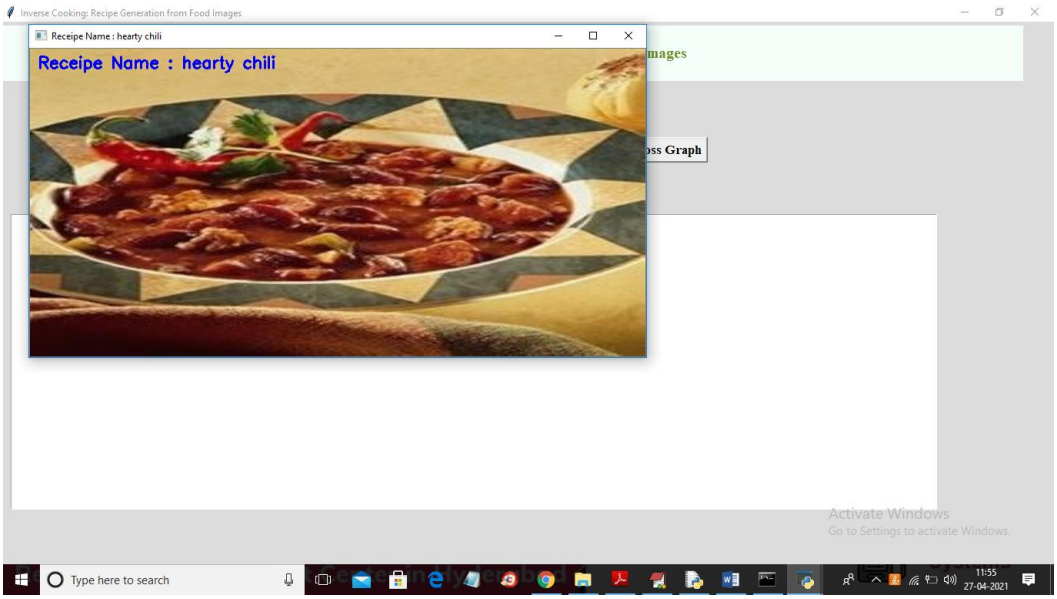
Screenshot 5.5 Model Summary

In above screen to train model we have filtered dataset with multiple layers and each layer use image with different size as 62 X 62, 31 X 31 etc. Now click on 'Upload Image & Predict Recipes' button to upload test images



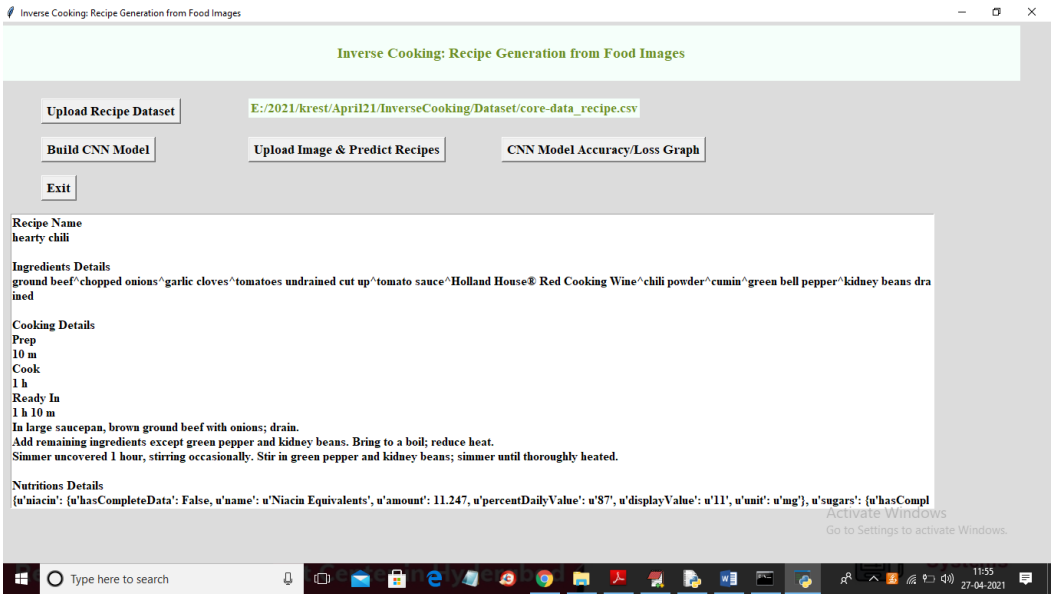
Screenshot 5.6 Image Upload

In above screen select any image and then click on 'Open' button to get below result



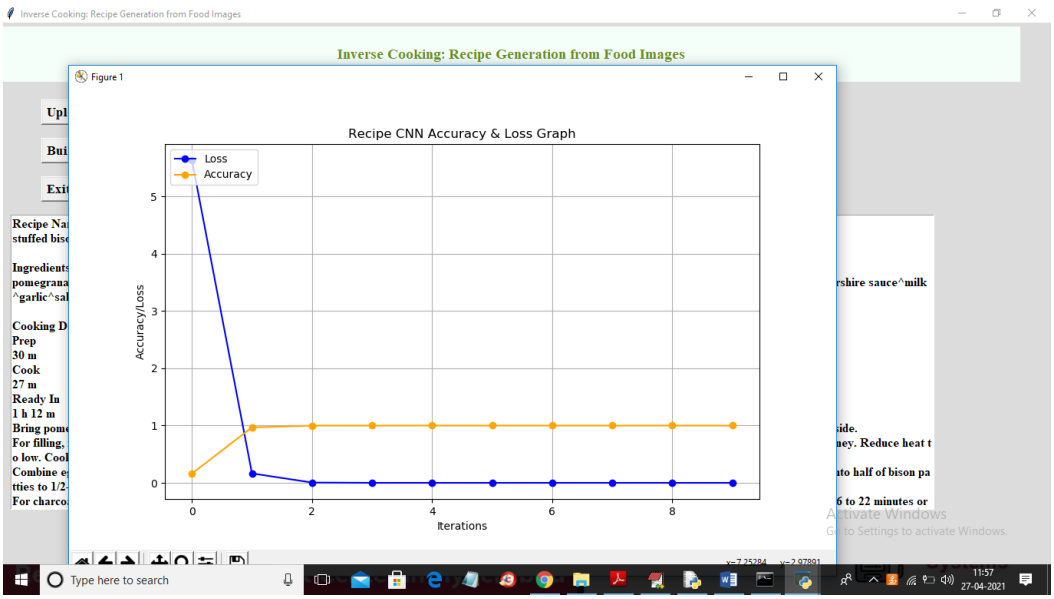
Screenshot 5.7 Recipe Predicted

In above screen uploaded image recipe identified as ‘hearty chili’ and now close above image to get below details



Screenshot 5.8 Recipe Summary

In above screen we can see recipe name, ingredients details, cooking and nutrition details and similarly you can upload any image and get recipe



Screenshotss 5.9 Graphical Analysisss

In above graph x-axis represents epochs and y-axis represents accuracy/loss value and blue line represents loss and orange line represents accuracy and in above graph with each increasing epoch accuracy got increase to 1 (100%) and loss decrease to 0. Any CNN model whose accuracy is high and loss is less will be consider as efficient model.

6.TESTING

6.TESTING

6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent.

Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 FUNCTIONALTESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes.

6.3 TESTCASES

6.3.1 UPLOADING DATASET

Test case ID	Test case name	Purpose	Test Case	Output
1	User uploads dataset	Use it for train the model	The user uploads the recipe dataset	Uploaded successfully
2	User uploads another dataset	Use it for train the model	The user uploads the a non-recipe dataset	Uploaded successfully

6.3.2 TEST CASES

Test Case ID	Test Case Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
T1	Check for Web Scrapping Images	Connect to the Internet to get the food images	URLs for Food Images to be Inputted	Food Names stored in folder are Printed	As Expected	Pass
T2	Check for Recipe Dataset loading	1. Upload Dataset	Recipe Dataset	Dataset loaded, Recipe's data loaded	As Expected	Pass
T3	Check for Model Accuracy	1. Upload Dataset 2. Build CNN Model	CNN Model with Encoders & Decoders	CNN training process completed with Accuracy = 99.66	As Expected	Pass
T4	Check for Food Image Prediction	1. Upload Dataset 2. Build CNN Model 3. Upload Image & Predict Recipes	Test Images of Indian and other food items to be given	Recipe Name, Ingredient Details, Cooking Details Printed	As Expected	Pass
T5	Check for Accuracy Graph	1. Upload Dataset 2. Build CNN Model 3. Upload Image & Predict Recipes 4. Accuracy/Loss Graph	Accuracy & Loss to be known	Graph of Accuracy vs Loss	As Expected	Pass
T6	Check for Exit	Exit	Final Step to close the application	Closes User Interface	As Expected	Pass
T7	Giving an unknown Food Image	1. Upload Dataset 2. Build CNN Model 3. Upload Image & Predict Recipes	Image of a Test Data	Recipe details of Spicy Indian Dahl	As Expected	Pass

7.CONCLUSION

7.CONCLUSION & FUTURESCOPE

7.1 PROJECT CONCLUSION

In the present thesis, we have presented a Food ingredient recognition and cooking Recipe prediction model. Cooking Recipes are quickly predicted and displayed upon inputting a food image obtained directly from web sources. This is a unique model that incorporates visual object detection into the Culinary recipe recommendation system. We identified that numerous Deep learning methodologies could be accompanied in this process namely:

- Convolutional Neural Network Algorithm,
- Sequential Encoder-Decoder architecture
- Data Visualization using Python libraries.

Based on an image of a dish, the proposed multi-layered model generates a relevant title to the food recipe, ingredients list, and detailed cooking directions along with time duration in textual format. Various studies have proven that if the model can pay attention to both image features and the ingredient properties while creating the output, the quality of the instruction set is be greatly maximized. The design and dataset curation of the proposed reverse cooking model and medical diagnostic image captioning systems from the literature survey were found to be equivalent. Our Image-to-Recipe translation model takes a food image and turns it into a recipe with a title, ingredients, and cooking directions in order. Generating recipe instructions implies making decisions based on features present in the food image. Finally, we compared our proposed system to traditional text-to-recipe retrieval methods using NLP.

We designed the model so that it can be easily used at cafes, restaurants grocery stores as well as at homes before cooking. Our project outcome indicates that the CNN architecture-driven model has yielded about 96.05% accuracy. We believe that further making use of larger and more diversified datasets containing varied cuisines prepared globally for Testing and Training will surely increase the scope and functionality of our model.

7.2 PROJECT FUTURESCOPE

The future scope in any Image recognition model would be boundless with changing times and ours being Food image recognition, which has a much broader scope for enhancement and addition of auxiliary features. Some of which are listed below:

Recipe-based Voice AI: This will work as a Virtual Assistant by vocalizing the entire recipe to the user, enabling them to listen to the recipe anywhere at any time and cook simultaneously. This model would allow users to concentrate on their cooking rather than figuring out how to unlock their mobiles with food-stained fingertips. Once you unlocked the mobile, the model will list out easy-to-follow steps in food preparation.	**Enhanced and detailed Dietary systems:** The addition of additional nutritional information along with health benefits will provide insights on effective health management and disease prevention among users. It is not only useful for goal setting, but it acts a tool for nutritionists who want to take a glance at their clients eating habits, dietary intake, and cooking procedures for a detailed analysis of the client's food lifestyle.
Alternatives Prediction of Ingredients: Diabetics, Cardiac, Gastrointestinal, and various other patients avoid few ingredients and prefer alternative which can be recommended by the model for each ingredient.	**Integration with Android applications:** To improve the scalability and portability of Image-based prediction systems, the existing model may be integrated into real-world mobile devices and cloud computing-based systems.

8.BIBILOGRAPHY

8. BIBILOGRAPHY

8.1REFERENCES

[1] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In ECCV, 2014.

[2] Micael Carvalho, R´emi Cad`ene, David Picard, Laure Soulier, Nicolas Thome, and Matthieu Cord. Cross-modal retrieval in the cooking context: Learning semantic text-image embeddings. In SIGIR, 2018.

[3] Jing-Jing Chen and Chong-Wah Ngo. Deep-based ingredient recognition for cooking recipe retrieval. In ACM Multimedia. ACM, 2016.

[4] Jing-Jing Chen, Chong-Wah Ngo, and Tat-Seng Chua. Cross-modal recipe retrieval with rich food attributes. In ACM Multimedia. ACM, 2017.

[5] Mei-Yun Chen, Yung-Hsiang Yang, Chia-Ju Ho, Shih-Han Wang, Shane-Ming Liu, Eugene Chang, Che-Hua Yeh, and Ming Ouhyoung. Automatic chinese food identification and quantity estimation. In SIGGRAPH Asia 2012 Technical Briefs, 2012.

[6] Xin Chen, Hua Zhou, and Liang Diao. Chinesefoodnet: A large-scale image dataset for chinese food recognition. CoRR, abs/1705.02743, 2017.

8.2 WEBSITES

1. https://www.longdom.org/peer-reviewed-journals/reputedartificialintelligencejournals-39160.html
2. https://research.fb.com/wp-content/uploads/2019/05/Inverse-Cooking-Recipe-Generation-from-Food-Images.pdf
3. https://www.edureka.co/blog/python-libraries/
4. http://pic2recipe.csail.mit.edu/im2recipe.pdf
5. https://www.researchgate.net/publication/341998176_Sentiment_Analysis_Based_on_Deep_Learning_A_Comparative_Study
6. http://vireo.cs.cityu.edu.hk/jingjing/papers/chen2016deep.pdf
7. https://www.researchgate.net/publication/266357771_Food_Detection_and_Recognition_Using_Convolutional_Neural_Network