

# FoodieCal: A Convolutional Neural Network Based Food Detection and Calorie Estimation System

by

Chowdhury Zerif Mashrafi

16301138

Shahriar Ahmed Ayon

16301209

Abir Bin Yousuf

16101044

Fahad Hossain

16301139

A thesis submitted to the Department of Computer Science and Engineering  
in partial fulfillment of the requirements for the degree of  
B.Sc in Computer Science

Department of Computer Science and Engineering

Brac University

January 2021

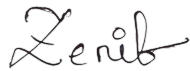
© 2021. Brac University  
All rights reserved.

# Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

## Student's Full Name & Signature:



---

Chowdhury Zerif Mashrafi  
16301138



---

Shahriar Ahmed Ayon  
16301209



---

Abir Bin Yousuf  
16101044



---

Fahad Hossain  
16301139

# Approval

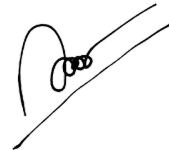
The thesis/project titled “A Convolutional Neural Network Based Food Detection and Calorie Estimation System” submitted by

1. Chowdhury Zerif Mashrafi (16301138)
2. Shahriar Ahmed Ayon (16301209)
3. Abir Bin Yousuf (16101044)
4. Fahad Hossain (16301139)

Of Fall, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 11, 2021.

## Examining Committee:

Supervisor:  
(Member)



---

Muhammad Iqbal Hossain, PhD  
Assistant Professor  
CSE Department  
BRAC University

Program Coordinator:  
(Member)



---

Md. Golam Rabiul Alam, PhD  
Assistant Professor  
CSE Department  
BRAC University

Head of Department:  
(Chair)

---

Mahbubul Alam Majumdar, PhD  
Professor and Chairperson  
Department of Computer Science and Engineering  
BRAC University

# Abstract

According to recent studies across the world, we can see that a healthy diet is the key to having a sound health and body. People nowadays are more concerned with their diets than ever before. With the advancement of science, it is now viable to construct a unique food identification system for keeping track of day to day calorie intake. However, building this kind of system creates several complications on constructing and implementing the model. In our paper, we have developed a new neural network based model which will predict the food items from a given image and show us the estimated calorie of the detected food items. In order to achieve our goal, we have prepared a dataset of around 23000 images for 23 different food categories. Initially, we have developed a single food detection system combining CNN max pooling and ResNet. From our experimentation, we have achieved 93.33% accuracy in this case. Furthermore, we have also taken a step forward to build a system which can detect multiple foods by training CNN with features extracted by Inception V3. We have achieved 89.48% accuracy for this model and we deployed both of our systems on a webpage. The user has to upload an image of food item in the webpage and our system will predict the food item along with the estimated calories in real time.

**Keywords:** Food Detection; CNN; ResNet; Inception V3

## **Dedication**

This research is dedicated to the people who are suffering from obesity and overweight related health hazards. We also want to dedicate it to people who are health conscious and willing to maintain a healthy and balanced calorie intake.

## **Acknowledgement**

First and foremost, we want to show our gratitude to Allah for helping us complete this work without any kind of major difficulties. Next, we want to thank our parents who supported us utmost in reaching where we are now. Furthermore, we want to thank our honorable Supervisor who helped us by giving continuous feedback and suggestions on improving our work and reaching the ultimate goal.

# Table of Contents

Declaration	i
Approval	ii
Abstract	iii
Dedication	iv
Acknowledgment	v
Table of Contents	vi
List of Tables	viii
Nomenclature	ix
<b>1 Introduction</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 Problem Statement . . . . .	2
1.3 Research Objective . . . . .	3
<b>2 Background</b>	<b>5</b>
2.1 Literature Review . . . . .	5
2.2 Convolutional Neural Network . . . . .	7
<b>3 Proposed Model</b>	<b>11</b>
3.1 Dataset Description . . . . .	11
3.1.1 Dataset Collection . . . . .	11
3.1.2 Dataset Sample . . . . .	12
3.1.3 Data Preprocessing . . . . .	12
3.1.4 Feature Extraction Techniques . . . . .	14
3.2 Model Description . . . . .	19
3.2.1 Model Description for Single Food Detector . . . . .	19
3.2.2 Model Description for Multiple Food Detector . . . . .	21
<b>4 Experimentation</b>	<b>23</b>
4.1 Single Food Detector . . . . .	23
4.2 Multiple Food Detector . . . . .	24

<b>5</b>	<b>Result Analysis</b>	<b>27</b>
5.1	Analysis on Single Food Detection . . . . .	27
5.2	Analysis on Multiple Food Detection . . . . .	30
5.3	Limitations . . . . .	34
<b>6</b>	<b>Conclusion and Future Work</b>	<b>35</b>
	<b>Bibliography</b>	<b>37</b>



# List of Tables

5.1	Prediction percentage for test image 1 . . . . .	28
5.2	Prediction percentage for test image 2 . . . . .	29
5.3	Prediction percentage for test image 3 . . . . .	31
5.4	Prediction percentage for test image 4 . . . . .	33
5.5	Output comparison for some test images . . . . .	34

# Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

*BoF* Bag of features

*CNN* Convolutional Neural Network

*MKL* Multiple Kernel Learning

*ReLU* Rectified Linear Unit

*SFTA* Segmented Fractal Textual Analysis

*SIFT* Scale Invariant Feature Transform

*SURF* Speed-Up-Robust-Features

*SVM* Support Vector Machine

*WHO* World Health Organization

*WISeR* Wide Slice Residual Network

# Chapter 1

## Introduction

### 1.1 Introduction

With the advent of technologies, nowadays, people have become more aware of what they are consuming to satisfy their hunger. This is due to the fact that obesity has become a common phenomenon in recent times. People, especially the young generation are suffering from obesity because of lack of awareness about what kind food they should consume. They are highly attracted to unhealthy junk foods and uncontrolled amounts of calorie intake which leads to a variety of chronic diseases like heart blockage, stroke, diabetes, liver problems, kidney damage etc.

The World Health Organization (WHO) has reported [19] that, in 2016, around 1.9 billion people (aged 18 years and above) around the world are suffering from overweight and 650 million among them are obese. 40 million children (under the age of 5) were overweight in 2018 which is an alarming sign for the world. In 2019, an estimated 38.2 million children under the age of 5 years were overweight or obese. Obesity was once considered a major problem only in the highly developed countries, but nowadays it has become a common scenario in lower and middle-income countries too, particularly in urban areas. In Africa, the number of overweight children under 5 has increased by nearly 24% since 2000. Almost half of the children under 5 who were overweight or obese in 2019 lived in Asia.

A recent study by National Cancer Institute [11] suggests that the fat tissue generated due to excessive calorie intake creates low-level inflammation which causes DNA damage and leads to cancer. Also, obese people have increased level of insulin in their blood that promotes Type-2 diabetes and eventually leads to colon, kidney, prostate and endometrial cancer[7].

To avoid these types of problems, people have now started to realize that they should be well aware about the side effects of foods and what their body needs in order to keep themselves healthy. So, food journaling has become popular to help people keep track of how much calorie they should consume in a particular time. It also lets people decide whether they are consuming food out of boredom or satisfying the hunger. But the journaling maintained in an analog way by writing on note may create monotony and it is pretty much time consuming. Besides, the youth nowadays don't like the idea of manually writing on notes and maintaining the food

journals. It sometimes becomes hectic and people may tend to forget what they had consumed earlier which they forgot to record in the food journal. So, there is a need to build a system which can keep track of the daily calorie intake in a digital way and recommend people about how much to eat and how much to avoid.

## 1.2 Problem Statement

The availability of smartphones all over the world has enabled people to take pictures of what they are eating. So there comes the concept of food detection using the smartphones or camera and journaling in a digital way. Researchers are trying to develop a system that does the food journaling automatically. That is by keeping track of user information and helping the user to decide what suits the body best. So, developing a system that identifies the food from an image and measures the calorie from it can help people by decreasing the rapid rise of diseases related to excessive calorie intake and obesity.

The people of Bangladesh are suffering from overweight and obesity problems for a long time. In 1980, 7% of adults and 3% of children were overweight or obese in Bangladesh. In 2013, those rates had climbed to 17% for adults but only 4.5% for children — according to The Institute for Health Metrics and Evaluation (IHME) of the University of Washington. The recent cases of obesity among Bangladeshi young generation is very high. Due to the influence of western lifestyle, the internet and advertisements on TV, the youth including children are more likely to consume fast food and other junk foods more than before which leads to varieties of major and minor health problems. It is alarmingly increasing in our country because of lack of checking on calorie intake and also unhealthy food schedules.

So, it has become a necessity to develop a system which can recommend a healthy and balanced diet. But it is not a simple task to accomplish. It requires identification of food items from the images captured, extracting features from the image, processing it and then calculating and showing calorie intake for the user to have an overall idea of how much calorie the user should take for a single day. It becomes a great challenge even if we relieve on the overall objective and only focus on food recognition from images captured. It presents a lot of uncertainty because there are lots of variations of different food images of different cuisines around the world. Even the appearance of the same food item may vary in different places around the country according to the availability of ingredients, taste of people coming from different regions and also the personal taste of the chef. The scientists have done years of research and provided us with various types of hypotheses. They provided theories with proof of identifying various food items using different types of algorithms such as machine learning, image processing, convolutional neural networks, deep learning, artificial intelligence etc. But every one of them has their own limitations. Most of the existing systems have a low accuracy in detecting food items let alone showing overall estimated calorie.

This has motivated us to step up and do an extensive research on this issue. We have developed a system that not only identifies the food items with a great accuracy

from an image but also shows the amount of calorie the food item contains. In this paper, we have proposed a model which is focused on Convolutional Neural Network (CNN) for food detection from a given image. Initially, we have developed a dataset of 23 food categories for identifying multiple food items. Then we extracted features from the images using Inception V3 and trained the model with CNN. Next, we deployed the model in a webpage which is for showing the result. In the end, the result is evaluated in the webpage when an image is given as an input and the system predicts what food it is along with showing the calorie amount. We believe the development of such a system may help especially the young generation fight obesity and overweight problems and lead to a happier and healthier life.

The rest of the paper is organized as follows - Chapter 2 evaluates the related works done previously on the food identification process and Chapter 3 describes the dataset that we preprocessed and also the feature extraction techniques we have used in our model. In Chapter 4, the whole experimentation process is described to have a clear view on how the model operates and in Chapter 5, the result of the model is analyzed to support our claim. Lastly, Chapter 6 concludes our work.

### 1.3 Research Objective

The main aim of this work is to classify food items using a combination of feature types for image feature extraction with CNN and recommend food to the user by estimating total calorie. In previous related works we encountered that there are many proposed models for recognizing single food items. But there is lack of systems that can detect multiple foods and also recommend the user by showing the amount of calorie the food contains. So, we have determined the following objectives of our work -

- The first objective of our proposed model is to develop a model that solves the idea of identifying multiple food items at a time based on the dataset we trained our model with. As a result, it will easily be able to detect multiple food items at a time from a single image.
- The second objective is to develop a dataset which we will feed our model for training and also use for validation. For this process, we have categorized 18413 images into 23 different classes of food items. These images are used to train the CNN model for recognizing the food item. Then another 4538 different images are used for the validation process.
- The third objective is to choose which feature extraction technique ensures greater efficiency and accuracy. For this, we have created a combination of feature selection techniques in our proposed model. For example: in our proposed model we have combined the Max Pooling and RESNET techniques for identifying the single food items initially. Later, we have gone for Inception V3 for identifying multiple food items which gives more accuracy in extracting features from images provided. It is seen that Segmented Fractal Textual Analysis (SFTA) , Lab Color Space, Bag of features (BoF), Speed-Up-Robust-Features (SURF) and other feature models show less accuracy while extracting features from images. Besides, some of the feature extraction techniques are

time consuming, less efficient and show less accuracy at times. As a result, we decided to implement Inception V3 to create a better model for feature extraction.

- Our fourth objective is to ensure greater efficiency and higher accuracy for food image classification. For this we trained our model with Convolutional Neural Networks (CNN). CNN takes little time in preprocessing compared to other image classification techniques and it provides more accurate results than any other techniques currently available.
- Last but not the least, our fifth objective is to estimate calories through our proposed model. After developing the model which can identify foods from images captured, it will calculate the estimated calorie from the image by comparing with the dataset of food calorie chart which we will feed into the system after classification approach. As a result, anyone can easily know the estimated calorie of the food.

# Chapter 2

## Background

### 2.1 Literature Review

Quite a few works have been done on the food recognition process using various types of techniques. But each one of them have their own limitations. We have gone through some of the earlier proposed models on Food identification and investigated them in detail. In the research paper [6], the researchers had taken pictures in two phases, one phase containing 300 pictures taken using smartphones and another phase containing 300 images taken using the Google Glass. In the second step, they used Geo-tagging and image localization techniques to obtain location. Obtaining location, they used API's of Yelp and Google Places for identifying if the images' geo-tags match with the geo-tag of a restaurant. After locating the restaurant, they collected the images of the food menu from Google which can be called as weakly-labelled training data. Next, they segmented the image of food from the surrounding image and collected as test data. For back end classification, they used MKL (Multiple Kernel Learning) and for feature extraction from the training and test data, they used Harris-Laplace point detector. As the image taken in restaurants may vary due to lighting conditions, they used 6 descriptors- 2 color based (Color Moment Invariants and Hue Histograms) and 4 SIFT (Scale Invariant Feature Transform) based (C-SIFT, Opponent SIFT, RGB-SIFT and SIFT). Then they performed codebook building for Bag of Words representation, kernel pre-computation and finally classification using SMO-MKL. After performing all that, they found out that American, Indian and Italian cuisines gave the best accuracy. But the accuracy for Mexican and Thai cuisines are limited. With this setup using 600 images of 5 different cuisines, they found the overall average accuracy of food identification is 63.33%. Their feature extraction techniques take time for processing and it is less efficient in terms of performance and accuracy. Besides, their model cannot identify multiple food items from a given image. Our model overcomes the deficiencies by using Inception V3 which takes really less time for processing and it can also detect multiple foods from an image.

In research work [13], Martinel et al. 2018 proposed a model that has two branches- a residual branch that encodes generic visual representation of food images and the vertical layers of the food dishes are captivated by a slice network branch. Features extracted from two network branches are being merged by the WISeR architecture. For validating the proposed model, they used three datasets of food images-

UECFOOD100, UECFOOD256 and Food-101. For experimentation, they did not train their residual branch from the very beginning as it required larger dataset with more images which was not available. They started residual branch training from WRN architecture which was pre-trained on ImageNet 2012 classification dataset. But the slice branch was trained from the scratch as the slicing and vertical images may vary and produce less accuracy. For the fine tuning of images, they cropped the images in a smaller dimension of 256 pixels. For testing, they considered standard 10 crop testing. They performed the model training by using stochastic gradient descent with mini-batches containing 24 samples. While comparing with the state-of-the-art approaches, they found out that their residual branch largely outperforms the slice branch. The residual branch for all the datasets got accuracy above 90% and the sliced branch got accuracy of around 60%. When the WISeR architecture was trained from scratch, they got accuracy of 78.12%, 68.37%, and 79.45% on the three considered datasets respectively. In this work, the researchers got a higher accuracy for the pre-trained residual branch but when it comes to the slice branch which they trained from the scratch, they got around 60% accuracy. Our CNN based model outperformed them by a big margin by getting an accuracy of 89.48% which was trained from the scratch.

In Subhi et al. 2018 research paper [15], the authors proposed a model which can identify food items based on CNN algorithm. They also used a new dataset for local Malaysian food which contains eleven food categories with (5800) images. They utilized two datasets for their proposed model. For food/non-food classification, they have used FOOD-101 dataset and for the grouping of food items, they have applied their proposed local Malaysian food dataset. Additionally, they claimed that they had performed very extensive convolutional networks (24 weight layers) for food image classification. They proposed a more compound model containing of multi convolutional layers blocks before the final fully connected layer. The results confirm the significance of network depth in training visual representations. In comparison to our developed system, we used 48 layered Inception V3 and 152 layered ResNet models for food detection. In addition, our dataset contained 23000 food images. Our system was trained to detect 23 categories of food items. The system also estimates the calorie of detected food items. Moreover, our system provided an accuracy of 89% while detecting food items. The detected food item can be single or multiple at times.

In the research paper by Joutou et al. 2009 [1] and Hoashi et al. 2010 [2] color, surface, slope and Filter highlights are extracted from food pictures and a partitioned classifier is prepared for each include. At last, all the classifiers are weighted combined with the different parts of algorithms. Through their developed system 61.3% and 62.5% accuracy is accomplished for 50 and 85 categories of Japanese foods utilizing 9 and 17 highlights (5-fold cross approval in 8500 pictures). This system is additionally giving lower precision in terms of distinguishing food items from a picture. The dataset contains an add-up to a picture of 8500 food items. On the other hand, our system is exceedingly energetic in anticipating food items as well as assessing calorie with a precision of 89.48%. Our framework was prepared with more powerfully shaped pictures and the dataset was about 23000 pictures of food items. Moreover, it is able to detect multiple food items from a single input



image containing different food items.

In Chen et al. 2012 [3], they address the issues of including descriptors within the nourishment distinguishing proof issue and introduce a preparatory approach for the amount estimation using depth data. Sparse coding is utilized within the Filter and Local twofold design highlight descriptors and these highlights combined with gabor and color highlights are utilized to speak to food items. A multi-label SVM classifier is prepared for each feature and these classifiers are combined with a multi-class Adaboost algorithm. For assessment, 50 categories of around the world food items are used and each category contains 100 photos from distinctive sources such as physically taken or from Web web collections. An overall accuracy of 68.3% is accomplished and victory at top-N candidates achieved 80.6%, 84.8%, and 90.9% precision appropriately when N equals 2, 3, and 5 in this way making portable application down to earth. After examining these we are able counter our framework with numerous subtle elements. To begin with, our framework was prepared with 23 classes and each class contained 1000 pictures. It gave a precision of 89.48% while validating the dataset. Additionally, our framework not only recognizes food items from a given picture but also predicts the calorie of the distinguished food items. In addition, our framework can anticipate multiple food items at a time.

In Yoshiyuki et al. 2013 [4], researchers developed a mobile app for real-time multiple food recognition and named it as FoodCam. Within the system, users had to draw a bounding circle over food items on the screen. Using the GrubCut-based segmentation method this bounding circle was adjusted on the food region. The bounding circle was segmented to regions. For each region, histograms of gradients and Fisher Vector encoded feature vectors of color histograms computed and passed into SVM. In the end SVM classifiers classified and predicted the food item. They obtained an accuracy rate of 51.9% on their tests. Their main limitation is that, while using the mobile app, the user has to draw a bounding circle over the food items. It is quite inefficient. Besides, the accuracy they achieved is quite low compared to our proposed system. In our system, we developed a web page where users upload food images and our system predicts the food along with the calorie. We have achieved 89.48% accuracy in our proposed model.

## 2.2 Convolutional Neural Network

CNN has added a new dimension to machine learning. In terms of image detection, no feature can come close to CNN. That is why we have used convolutional neural network for object or image detection. CNN can take a set of images of an object, give the objects specific weights and biases and with these variables and can differentiate each object or image by following a set of steps in Figure 2.1

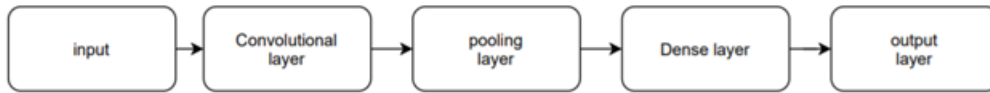


Figure 2.1: Steps of CNN [20]

Convolutional neural network mainly takes an image as an input and transforms it into a matrix with a default size. For a basic image it is basically simple to perform but for a complex image, for example 1080p(1920x1080) dimension images, the CNN needs to reduce the size of the matrix at the same time without losing the main features (edges, patterns etc.) of the images so that the prediction stay accurate [14] [16].

Now the images need a convolutional feature. For getting a convoluted feature from the matrix sized image the algorithm uses a Kernel Matrix from where we can get the convoluted feature [14].

While taking the image as an input, CNN uses some algorithms or methods such as max pooling which selects the highest value from the matrix which is helpful for images with highlighted features. There is another method for reducing the size of images which is called average pooling which mainly collects the average value of each pixel from the images. Average pooling is not widely used as max pooling. Average pooling takes the average value from the matrix so the prediction is average as well. So, we can say that max pooling performance is significantly better than the average pooling [14] [16].

CNN mainly works in multiple layers where the first layer takes the shape of the image as we have mentioned earlier. The output of this first layer will work as an input for the second layer and the output of the second layer will work as an input for the third layer and so on. This process will continue to perform until the last and final layer. In this way, the first convolutional neural network is responsible for extracting low level features (edges, gradient orientation etc.) of the images and the deeper layers are responsible for extracting the high-level features from the images [16]. For extracting high level features from complex images, CNN uses a fully connected network. After making the images suitable for multi-layer perceptron, CNN now flattens (Figure 2.2) the image into column vectors. These flattened images are put into a neural network which we call feed forward network and then we use back propagation for backtracking. For analyzing and finding errors, the process is repeated in every iteration while training [14] [16].

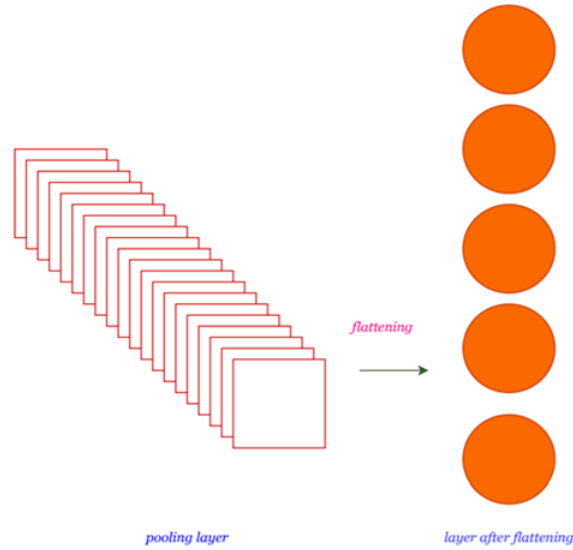


Figure 2.2: Flattening Process[22]

Feed-forward neural networks are a kind of artificial neural network where the relation between units does not perform a repetitive cycle. Feed forward is one of the first one of its kind for image detection where it is much simpler than recurrent neural network. The name feed-forward is mainly because it only forwards the data to the next layer and since it has no loops it cannot be traversed backwards. Feed forward network mainly takes a function  $f$  on a default size of inputs  $i$  so that  $f(i)=j$  where we can say the training pairs  $(i, j)$  [17] which we can see in (Figure 2.3).

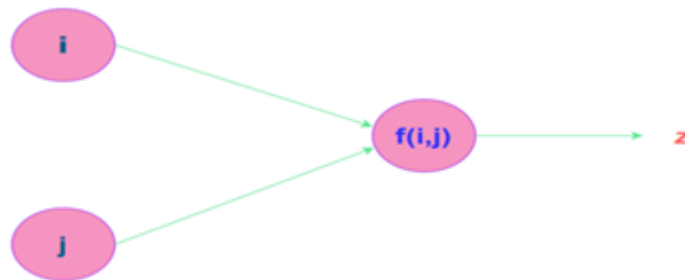


Figure 2.3: Feed forwarding network [12]

After feed forwarding we need back-propagation to traverse backwards to the previous layer. Back propagation is easy and simple to perform. Back propagation mainly uses inputs and weights for calculating every neuron from the input layer and calculates the error from the previous layer. After calculating the error, Back propagation traverses back to the hidden layer or inner layer for adjusting the weight in order to decrease the error. Back propagation keeps repeating this process until the algorithm reaches the potential output. There are mainly two kinds of back propagation and they are- Static back propagation and Recurrent back propagation [21].

Static back propagation mainly works on static inputs and static outputs, so it is mostly used for static image recognitions. On the other hand, recurrent back propagation is mainly used for non-static inputs and outputs. Back propagation takes full use of the chains of fully connected nodes and networks by traversing back to hidden layers and it can work with any number of nodes and inputs [21] which is shown in Figure 2.4

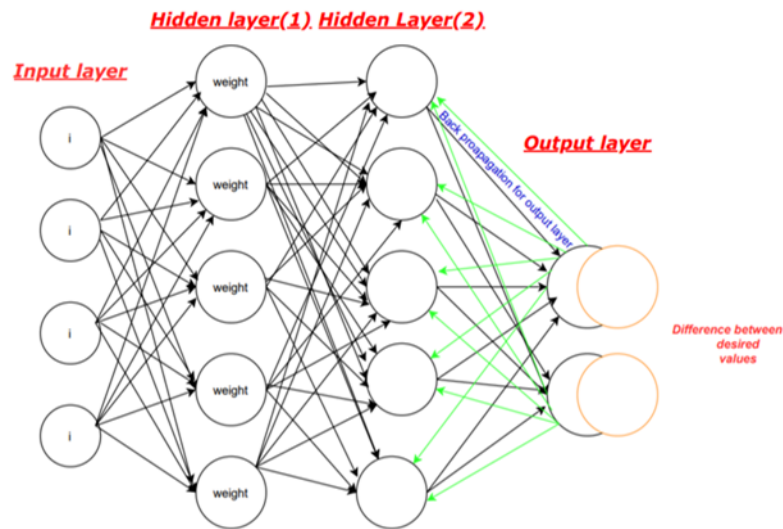


Figure 2.4: Back Propagation Process [21]

# Chapter 3

## Proposed Model

### 3.1 Dataset Description

#### 3.1.1 Dataset Collection

**Single Food Detector:** Initially, we made a food dataset with 20 different food items. We managed to put together 1000 images per food item and started to work with our single food detector system. There were total 20000 images in our dataset and we made sure that all of these images are put in the right categories. The 20 food classes we used for our dataset are:

- Chicken Wings, Chocolate Cake, Churros, Cupcakes, Deviled Eggs
- Donuts, French Fries, Fried Rice, Grilled Cheese Sandwich, Hamburger
- Hot and Sour Soup, Hotdog, Macarons, Oysters, Pancakes
- Pizza, Samosa, Seaweed Salad, Steak, Waffles

**Multiple Food Detector:** For constructing the dataset for multiple food detector, we used the same 19 categories as the Single food detector. We did not use French Fries for the multiple food detection and instead, we added 4 new categories in the dataset. Around 23000 images from 23 different food classes were used in constructing this dataset. Again, we made sure that all the food classes had the correct images in them before training our algorithm. Both Training Dataset and Validation dataset were thoroughly checked in both of our datasets (Single and Multiple food). The 23 food classes we used for our dataset are:

- Chicken Wings, Chocolate Cake, Churros, Cupcakes, Deviled Eggs
- Donuts, Fried Rice, Grilled Cheese Sandwich, Hamburger, Hotdog
- Hot and Sour Soup, Ice cream, Lobster Bisque, Macarons, Oysters, Pancakes
- Pizza, Samosa, Seaweed Salad, Steak, Waffles, Tacos, Tuna Tartare

### 3.1.2 Dataset Sample

We have used total 23 different food categories for making both of our datasets and each food class contains 1000 images. As a result, it is very difficult to show a good sample from the database. We tried to show at least 80 to 90 images from our dataset and we managed to put together 90 images in total in Figure 3.1. This image given below contains at least 3 to 4 images from each food category and it is visible from this sample image that we used various types and calories of food for constructing our dataset. In case of single food detector, we expect our system to work in such a way that it can detect the name of these foods and predict their calories correctly. We want our system to predict these food items from any input image containing these items. On the other hand, our multiple food detector is expected to predict all the food categories that can be seen in a single input image. If there are 2 or 3 food classes in an image, our system will be able predict all of them and with good accuracy. So, after constructing our datasets, we focused on preprocessing our data.

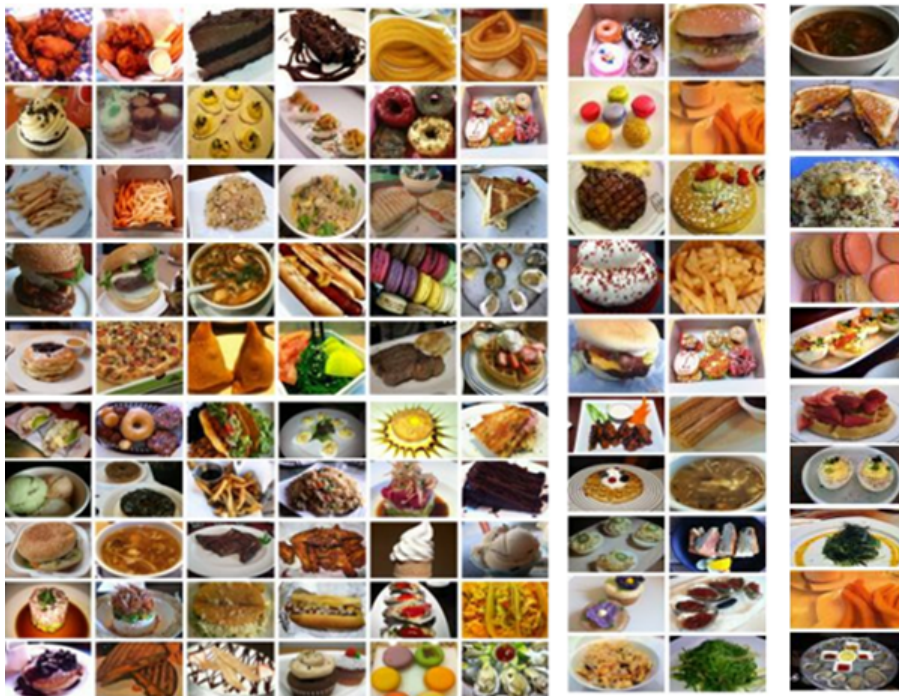


Figure 3.1: Sample Images from the Dataset

### 3.1.3 Data Preprocessing

#### Single Food Detector

For the dataset we used for detecting single items from an image, we had to import some necessary libraries, initialize directory and resize the images before extracting features. These are the libraries we had to import in order to make our dataset compatible with our model:

- os

- tensorflow
- numpy as np
- ImageDataGenerator

After importing these libraries, we initialized the directory of our dataset so that the model can get the images and start resizing them and after that start extracting features. We uploaded our whole dataset in the Google Drive and mounted and copied the path in our code. We made a folder named 20 food predict and we had a new folder there. In that new folder, there were 2 folders named Train and Validation. We had to initialize the directories of these 2 folders later in the code.

After locating the dataset directory our model started extracting features and for that, all of the images had to be reshaped as image height = 224 and image width = 224.

This is the whole Data Preprocessing process that was done so that our model can start extracting features from the images in our dataset.

## Multiple Food Detector

For the dataset we used for detecting multiple items from an image, we had to follow the same steps as before which are: importing some necessary libraries, initializing directory and resizing the images before extracting features. These are the libraries we had to import:

- os
- tensorflow
- numpy as np
- glob
- matplotlib.pyplot as plt etc.

After importing these libraries, we initialized the directory of our dataset so that the model can get the images and start resizing them and after that start extracting features. We uploaded our whole dataset in the Google Drive and mounted and copied the path in our code. We made a folder named 23 food predict and we had a folder named Multiple Food Detector there. In that folder, there were 2 folders named Train and Validation. We had to initialize the directories of these 2 folders later in the code.

Lastly, our images were resized so that our model can start extracting features and the images were reshaped as Image height=299, Image Width= 299 and Number of Classes = 23. After resizing the images, we proceeded to extract features using our inception v3 model.

### 3.1.4 Feature Extraction Techniques

#### Residual Network (ResNet)

Residual Network (ResNet) is a type of deep convolutional network which was first introduced by He et al. (2016) [9]. It was introduced to solve the “vanishing gradient” problem which was a barrier for achieving greater accuracy in image recognition. The fundamental concept of ResNet can be described as follows –

To solve the complex problems and achieve great accuracy, we pile up additional layers in a convolutional network. For example, for any kind of image recognition, the first layer may identify the edges from the image, the second layer may detect the color variant, the third layer may identify what sort of objects are present in the image and so on. But there is a limit on how many layers we can stack in the network because after a certain threshold, the error rate increases and the performance degrades. We can show this using a graph [9] shown in Figure 3.2

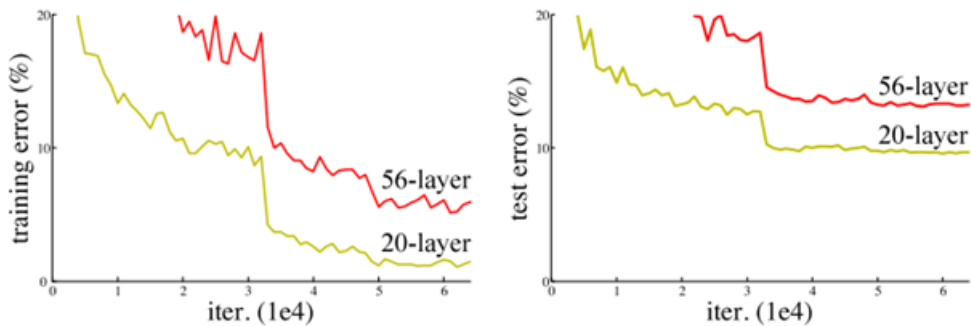


Figure 3.2: Error Percentage for increased layers on Convolutional Network

From the left graph we can see that the percentage of errors occurred while training images with 56 layers is relatively higher compared to using 20 layers. The graph on the right shows error percentage which also indicates that the deeper the layer goes, the higher the error rate becomes.

Here comes the concept of Residual Network (ResNet) which was introduced to train and test thousands of convolutional layers with a strong performance. In this concept, the authors [9] have used a term called Identity Shortcut Connection or Skip Connection. This skip connection basically figures out which layers degrade the accuracy and skip those in training. It collects all the layers that have no significance (also known as Identity Mappings) in training and skips over them. As a result, it compresses the model into fewer layers which enables the model to train faster. The authors [9] have allowed the network to learn residual mapping instead of letting the network learn underlying mapping. The residual blocks consist of layers. For example, ResNet-50 has 50 layers of residual blocks (Figure 3.3). These blocks pile up on top of each other to form the residual network. The authors [9] have argued that the stacking of layers on top of each other does not degrade the accuracy and



performance of the model because only the identity mappings are stacked here and the rest of the model works the same. As the irrelevant layers are skipped, it improves the performance of the model. In the proposed residual network model, the authors have denoted the underlying mapping as  $H(x)$  and let the network fit residual mapping which gives  $H(x) := f(x) - x$ . Finally, the original mapping is casted into  $H(x) := f(x) + x$ . Here,  $F(x)$  is denoted as the residual function. The authors [9] have claimed that it is easier to get to the solution of optimization of residual mapping  $F(x)$  than optimizing the underlying mapping  $H(x)$ . The structure of the residual block is shown below:

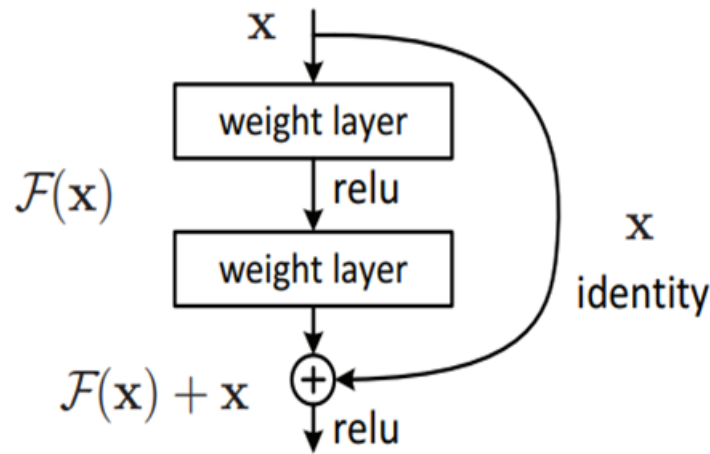


Figure 3.3: Residual Block [9]

The main architecture of ResNet (Figure 3.4) is inspired by the design of VGGNet-19. The VGGNet is a type of convolutional neural network that has introduced the importance of depth of the network. It basically consisted of 19 layers with mostly 3X3 filters. This architecture is combined with the identity shortcut connections to transform the model into the residual network. The residual network uses global average pooling which is similar to the one used in GoogLeNet. It was learned with a network depth of upto 152 layers which is known as ResNet-152. The researchers have found out that the ResNet shows better accuracy and it is computationally more efficient than GoogLeNet and VGGNet.

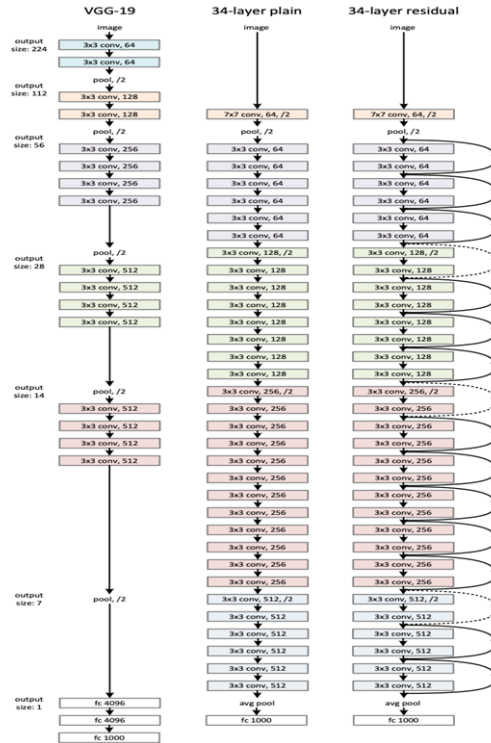


Figure 3.4: From the Left: the VGG-19 model [5], in the Middle: the plain 34-layer network and in the Right: Residual Network with 34 layers [9]

## Inception V3

Inception V3 was first introduced in a research paper developed by Szegedy et al.[10]. It was developed with a goal of modifying the earlier version of Inception architectures by reducing the computational complexity. In several biomedical applications GoogLeNet [8] achieved tremendous classification performance and Inception V3 turns out to be an extended version of this network. It is a 48 layered network architecture. In terms of computational efficiency, Inception Networks have outrun the VGGNet. A change is made in an Inception Network only when it is confirmed that computational advantages are not lost. Several techniques including factorized convolutions, regularization, dimension reduction etc. are applied to optimize the network and in an Inception v3 model. This results in losing the constraints and obtaining an easier model.

### Inception architecture:

**1. Factorized Convolutions:** It helps reducing computational efficiency by reducing the number of parameters involved in a network. In addition, It keeps a check on organize proficiency.

**2. Smaller Convolutions:** For faster training, it replaces bigger convolutions with smaller convolutions. For example: A “5 X 5” convolution having 25 parameters, replacing it by two “3 X 3” filters has only 18(3x3+3x3) parameters.

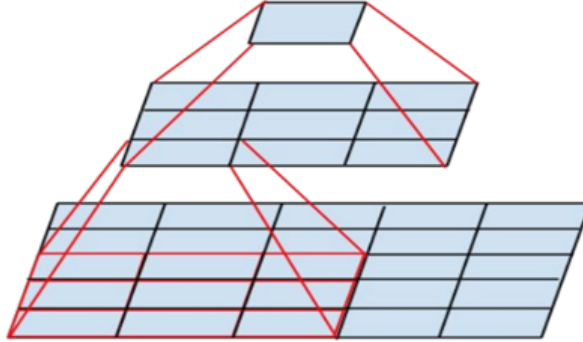


Figure 3.5: Replacing to smaller convolution. Recreated from [18]

Analyzing the figure, we can see that a 3x3 convolution in the middle and a fully-connected layer is fully below it. As both 3x3 are seen to be sharing weights among themselves, total number of computation can easily be reduced.

**3. Asymmetric convolutions:** It is seen that a “3 x 3” can be replaced by 1x3 convolution and it is followed by a 3x1 convolution. If we replace a 3x3 convolution by a 2x2 convolution, total number of parameters are found higher than proposed asymmetric convolution.

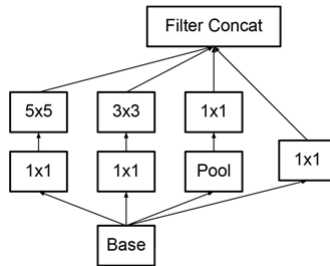


Figure 3.6: Basic Inception module. Recreated from [8]

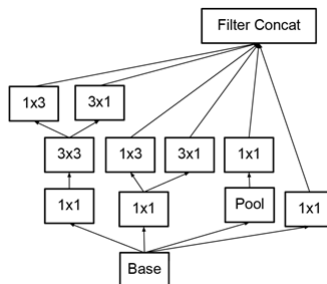


Figure 3.7: Expanded Inception module with filter blank. Recreated from [8]

**4. Auxiliary Classifier:** During training, an auxiliary classifier is seen to be inserted between layers. The loss incurred here is also added to main network loss. In Inception v3, the working mechanism of an auxiliary classifier is to work as a regularizer.

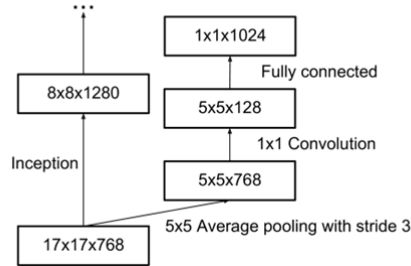


Figure 3.8: Auxiliary Classifier. Recreated from [18]

**5. Grid size reduction:** Pooling operation results in grid reduction. Besides, a more efficient techniques is proposed to combat the bottlenecks of computational cost.

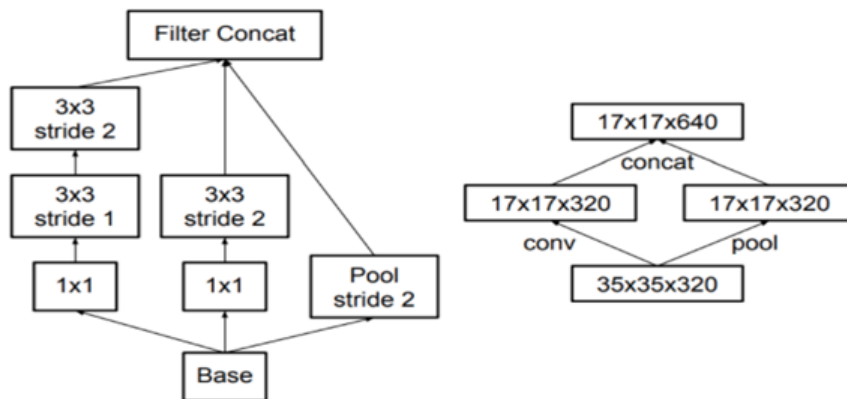


Figure 3.9: Grid size reduction. Recreated from [18]

Finally we get the complete architecture of Inception v3 utilizing all these concepts.

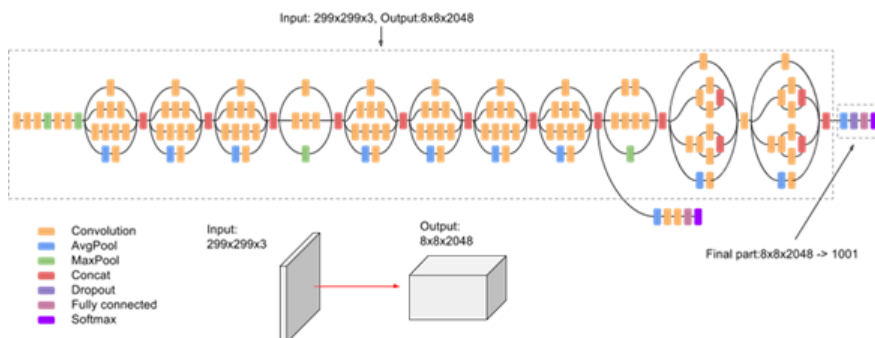


Figure 3.10: Inception V3 (Complete Architecture). Recreated from [10]

Key features of Inception V3 model:

1. Use of RMSprop:

- Gradient based optimization technique used in neural network training
- Normalizes gradient by moving average of squared gradient
- Uses adaptive learning rates instead of treating as a hypermeter
- Change in learning rate over time

2. Use of Batch Normalization:

- Standardizes the inputs to a layer for each mini-batch
- Significantly affects the system by stabilizing the learning process
- Training epochs being reduced dramatically while training
- Sets the mean and standard deviation of inputs for the layer as mean observed during training
- Can be modified into Batch Renormalization

3. Use of  $7 \times 7$  factorized Convolution

## 3.2 Model Description

### 3.2.1 Model Description for Single Food Detector

The whole workflow of our work can be described using a work-flow diagram. From this work-flow diagram shown in Figure 3.11, we can provide an overview of our work and all of the steps we followed to build our system. This is the model we used to build up the single food item detector:

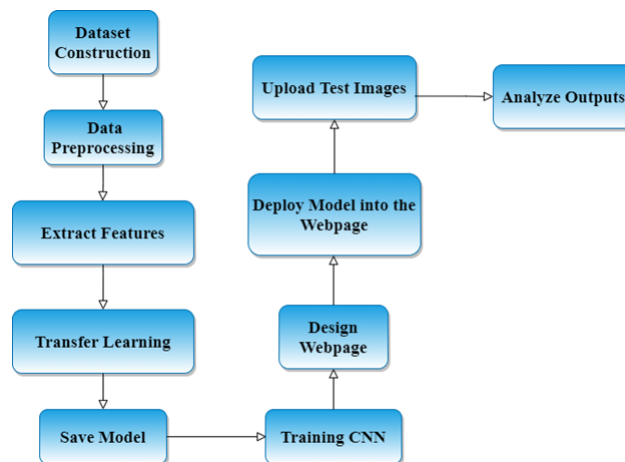


Figure 3.11: Work-Flow Diagram of the Proposed System

**Constructing Dataset:** Firstly, when we were building the single food detector, we wanted to work with 20 different food categories. We collected around 20000 images for our dataset (1000 images per food category) and divided them into Training dataset and Validation dataset.

- Exactly used 15097 images belonging 20 classes in the Training dataset
- Exactly used 4223 images belonging to 20 classes for the Validation dataset

**Data Preprocessing:** In order to make a dataset compatible with the system, some steps need to be followed. So, we had to import necessary libraries, initialize directory, resize all the images in our dataset before extracting features, standardize the data and assign different classes to different categories of food. These are the steps that we followed for preprocessing our entire dataset.

**Extracting Features:** After we used the Average pooling layer of CNN to identify unique features from all the images in our dataset and these features were used to train the CNN.

**Transfer Learning:** After extracting features from all the images in our dataset, we started building the ResNet152V2 model. We used ResNet152 version 2 for our ResNet model and used imagenet as weights. We will keep the extracted features from the 20 classes of our dataset in the in the last layer of the pre-trained ResNet model to get better prediction accuracy. This whole process is called Transfer Learning.

dropout_1 (Dropout)	(None, 512)	0	dense_1[0][0]
dense_2 (Dense)	(None, 20)	10260	dropout_1[0][0]
=====			
Total params: 59,653,652			
Trainable params: 6,841,364			
Non-trainable params: 52,812,288			

Figure 3.12: Last Dense Layer of ResNet Model

**Training CNN:** After finalizing required models, we proceeded to train the CNN (Convolutional Neural Network). We fed the features to CNN which were extracted using ResNet152V2 model. We tried different epochs for training but the final epoch and batch size we used Batch size = 32, Image height = 224, Image weight = 224, Number of classes =20, Epochs = 30.

**Save Model:** We saved our model after the training was done and then started working on our webpage. We saved our training code as 20f-predicts in our local directory.

**Designing Webpage:** We designed our webpage as our requirement. We wanted our system to take an image as input and give us the food name and food calorie as output. So we kept an upload image portion in our webpage and then we set the output to be shown right beside the uploaded image.

**Deploying Model into the Webpage:** After designing the webpage, we imported necessary libraries such as sys, os, re, tensorflow, flask, glob and we also had to write the prediction output code here for all the food categories. Then we loaded our trained model (model/my\_h5\_model.h5) in the deployment app.py code.

**Uploading Images:** After deploying the model, we collected some test images to see if our model is predicting right or not. Then we started uploading images into our webpage and started analyzing the output. We collected the testing images mostly from the food groups in Facebook, Google and some of the images were taken from our personal mobile devices as well.

**Analyzing Outputs:** After uploading an image, we have to click a button named predict and the webpage showed us the name of the food that was predicted from the uploaded image along with the calorie count. After getting the output, we had to analyze if the prediction was correct or not, the percentage of the given prediction.

### 3.2.2 Model Description for Multiple Food Detector

The whole workflow of our work can be described using a work-flow diagram. From this work-flow diagram shown in Figure 3.13, we can provide an overview of our work and all of the steps we followed to build our system. These are the major procedures of our code and even if some minor procedures are not shown here, they are included inside these major procedures.

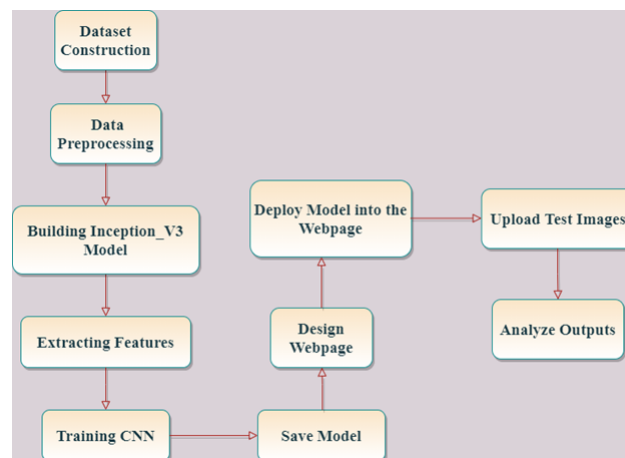


Figure 3.13: Work-Flow Diagram of the Proposed System

**Constructing Dataset:** Firstly, when we were building the multiple food detector, we wanted to increase food categories and so we decided to add 3 more food categories with the previous dataset (23 classes in total). We collected around 23000 images for our dataset and again divided them into Training Dataset and Validation Dataset as before. We used exactly 18413 images belonging to 23 classes for Training Dataset and exactly 4538 images belonging to 23 classes for Validation Dataset.

**Data Preprocessing:** In order to make a dataset compatible with the system, some steps need to be followed. So, we had to import necessary libraries, initialize directory, resize all the images in our dataset before extracting features, standardize

the data and assign different classes to different categories of food. These are the steps that we followed for preprocessing our entire dataset.

**Building Inception\_V3 Model:** After finishing data preprocessing, we focused on building the inception\_v3 model which will be used to extract features from the images.

**Extracting Features:** After building the inception\_v3 model, we used the model to assign individual weight to all the images in our dataset and these weights were used to train the CNN. The weights were saved in a file named inception\_model.hdf5.

**Training CNN:** After finalizing required models, we proceeded to train the CNN (Convolutional Neural Network). We fed the features to CNN which were extracted using inception\_v3 model. We tried different epochs for training but the final epoch that we used:

- Epochs = 12, Batch Size= 16
- Train sample = 16046, Validation Sample= 3959

**Save Model:** We saved our model after the training was done and then started working on our webpage. We saved all of our training, visualize and inception\_v3 code in the train folder.

**Designing Webpage:** We designed our webpage as our requirement. We wanted our system to take an image as input and give us the food name and food calorie as output. So we kept an upload image portion in our webpage and then we set the output to be shown right beside the uploaded image.

**Deploying Model into the Webpage:** After designing the webpage, we imported necessary libraries such as tensorflow, flask, glob, prettytable and we also had to initialize the food categories and the respective food calories for those categories. Then we loaded our trained model in the deployment app.py code. **Uploading Images:** After deploying the model, we collected some test images to see if our model is predicting right or not. Then we started uploading images into our webpage and started analyzing the output.

**Analyzing Outputs:** After uploading an image, we have to click a button named predict and the webpage showed us the name of the food/foods that were predicted from the uploaded image along with the calorie count and prediction accuracy. We also wrote the code in such way so that we get a table of prediction chart for every image we upload. This chart is shown in the command prompt.



# Chapter 4

## Experimentation

### 4.1 Single Food Detector

Firstly, we focused on building a system that will be able to detect one food item from an image and also show the calorie count for that particular food item. As we mentioned earlier, we used Max-pooling layer from CNN to extract features from our dataset images and we stored these features or weights of the images into our pre-trained ResNet model. After saving the weights in the model, we started training our CNN using to that model. We used activation = relu for the first two Dense layers and for the output Dense layer, we used activation = softmax for units = 20.

Then, we initialized our ResNet model and then we used the dense layer for class prediction. We used dropout to prevent our model from over-fitting. We used ReLu activation function for both of our dense layers and finally, for initializing the output, we used softmax activation function for our last layer. After initializing the activation functions, we initialized loss function as categorical crossentropy and we used learning rate = 0.001. Then we used accuracy as metrics.

Categorical Cross-Entropy loss is used to train a CNN to provide output as prediction percentages for all the images given as input. It is mainly used for multi-class classification. We used ReduceLRonPlateau and reduced the learning rate to 0.0001 after 19th epoch as our system was not learning properly. As we mentioned before, we used epochs=30 and batch size=32 for training our CNN. After the training was done for all 30 epochs, we got the following result from our algorithm shown in Figure 4.1:

```
Epoch 27/30
471/471 [-----] - 272s 577ms/step - loss: 0.2464 - accuracy: 0.9239 - val_loss: 0.7880 - val_accuracy: 0.8132
Epoch 28/30
471/471 [-----] - 272s 577ms/step - loss: 0.2347 - accuracy: 0.9271 - val_loss: 0.7870 - val_accuracy: 0.8144

Epoch 00028: ReduceLRonPlateau reducing learning rate to 0.0001.
Epoch 29/30
471/471 [-----] - 272s 578ms/step - loss: 0.2217 - accuracy: 0.9323 - val_loss: 0.8213 - val_accuracy: 0.8077
Epoch 30/30
471/471 [-----] - 272s 578ms/step - loss: 0.2128 - accuracy: 0.9333 - val_loss: 0.8388 - val_accuracy: 0.8061
```

Figure 4.1: Train loss, Train accuracy, Validation loss, Validation accuracy

From the figure 4.1, we can see that we got 93% accuracy and 80.61% validation

accuracy for 30 epochs. We can also see that, as the epoch count goes up, the loss percentage goes down for training. Moreover, as the epoch count goes up, accuracy of our training also goes up. As we put fewer images for validation dataset, the change in validation accuracy and validation loss are not as consistent as training accuracy and training loss.

We wanted to get at least 90% training accuracy from our system and we got 93.33%. As a result, we saved this trained model and worked with this saved model in our webpage. We linked our model to our webpage as followed: Model Path='model/my\_model.h5' Model= load model (Model Path)

After deploying this model, we tested our webpage by uploading some test images and started analyzing the outputs.

## 4.2 Multiple Food Detector

We now focus on detecting multiple food in an image. For this we build a model using inception\_v3 model to get higher accuracy for multiple food image. The model extracted features from all the images in the dataset and saved them as weights. Then we used these saved weights in the inception\_v3 model to train our CNN.

Firstly, we initialized our inception\_V3 model and then we used the dense layer for class prediction. We again used a dropout function to prevent our model from overfitting. We used ReLu activation function for both of our dense layers and finally, for predicting the output, we used sigmoid activation function for our last layer because we are using categorical crossentropy for our algorithm. Finally, we used learning rate=0.0001 and momentum=0.9.

After initialization we have used learning rate=0.0001 and momentum=0.9 and have used accuracy metrics. Then we started training our CNN according to our saved model. This time we have used 12 epochs and batch size=16. We got the following result shown in Figure 4.2:

```
Epoch 10/12
1002/1002 [=====] - 353s 352ms/step - loss: 0.6132 - accuracy: 0.8806 - val_loss: 0.5554 - val_accuracy: 0.8963

Epoch 00010: val_loss improved from 0.56445 to 0.55536, saving model to models\inception_model.hdf5
Epoch 11/12
1002/1002 [=====] - 351s 350ms/step - loss: 0.5866 - accuracy: 0.8880 - val_loss: 0.5413 - val_accuracy: 0.8957

Epoch 00011: val_loss improved from 0.55536 to 0.54126, saving model to models\inception_model.hdf5
Epoch 12/12
1002/1002 [=====] - 349s 348ms/step - loss: 0.5591 - accuracy: 0.8948 - val_loss: 0.5425 - val_accuracy: 0.8993

Epoch 00012: val_loss did not improve from 0.54126
Saving Model !
```

Figure 4.2: Train loss, Train accuracy, Validation loss, Validation accuracy

From the above figure we can see that for 12 epochs we got 89.48% accuracy and 89.93% validation accuracy. Moreover, as the epoch count goes up, accuracy of our training also goes up. As we put fewer images for validation dataset, the change in validation accuracy and validation loss are not as consistent as training accuracy

and training loss. We can show the changes in loss and accuracy with respect to epochs as these following 2 graphs in Figure 4.3 and Figure 4.4:

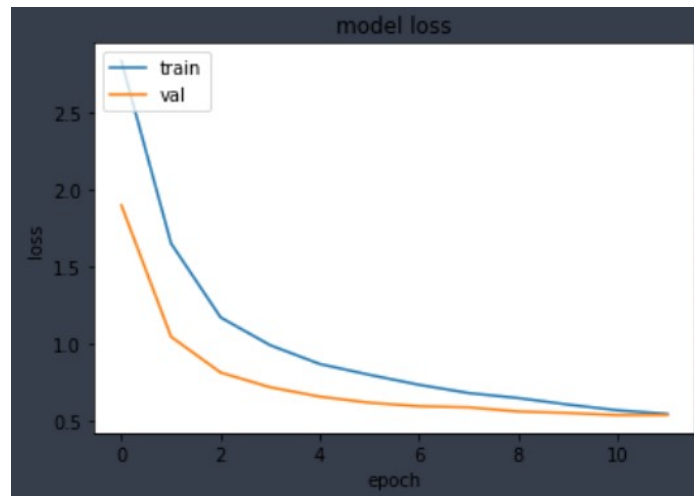


Figure 4.3: Changes in loss with respect to epochs

From the above figure, we can see that as epoch count goes up, training loss goes down and the validation loss goes down as well. At epoch=10, loss is 61.32% and validation loss is at 55.54%. When the epoch goes to 11, loss comes down to 58.66% and validation loss goes down to 54.13%. Finally, when the epoch is at 12, loss percentage is the lowest and it is at 55.91% and the validation loss goes down to 54.25%. This is how the loss and validation loss goes down with respect to increasing epochs. The loss and validation loss are at their highest at the start of the training and they are at their lowest percentage in the last epoch.

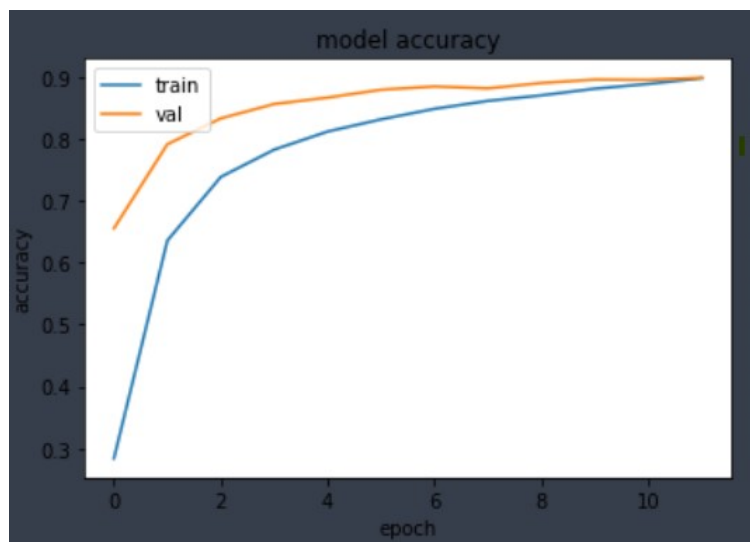


Figure 4.4: Changes in accuracy with respect to epochs

From the above figure, we can see that as epoch count goes up, training accuracy goes up and the validation accuracy goes up as well. At epoch=10, accuracy is 88.06% and validation accuracy is at 89.63%. When the epoch goes to 11, accuracy goes up to 88.80% and validation accuracy goes down to 89.57%. Finally, when the

epoch is at 12, accuracy percentage is the highest and it is at 89.48% and the validation accuracy goes up to 89.93%. This is how the accuracy and validation accuracy go up with respect to increase in epochs. The accuracy and validation accuracy are at their highest at the last epoch and they are at their lowest percentage in the starting epoch.

After the training was done, we saved our model and initialized our saved model in our webpage. We linked our webpage with our model as followed:

After finishing the code for deployment, we opened our webpage in our browser and started uploading test images and analyzing our outputs. We designed our webpage in a way that we can upload an image and get the prediction for that image in real time. The design of our webpage is as shown in the Figure 4.5:

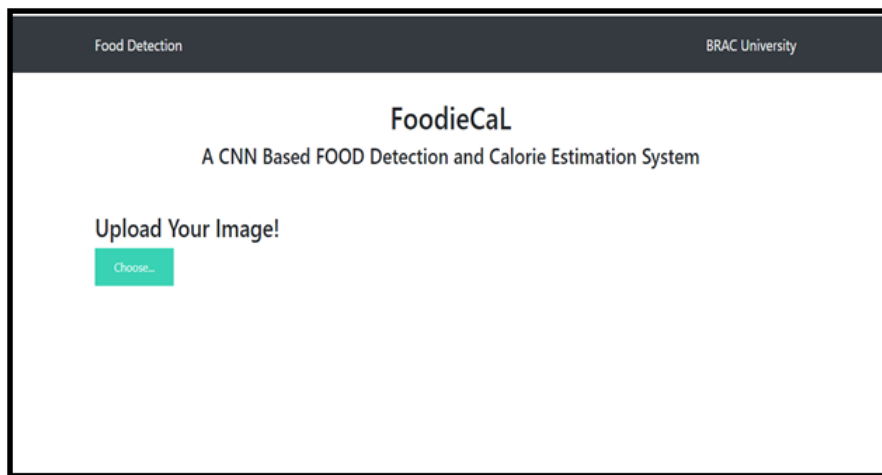


Figure 4.5: Webpage Layout

These were the experiments that we conducted with our systems (Single and Multiple food detectors). At first, we build the system in a way so that it can detect single food item from an input image. After we have seen that our system was able to detect food items successfully, we wanted to change our model so that our system can detect multiple food items from a single image. So we started using inception v3 model and finally, we were able to make a complete Food Detection System.

# Chapter 5

## Result Analysis

### 5.1 Analysis on Single Food Detection

After designing the webpage, we started to test our webpage by uploading some images and analyzing the outputs. Firstly, we had to click the choose button and it took us to our local directory. We uploaded an image of deviled eggs and uploaded the image shown in Figure 5.1 into our webpage.



Figure 5.1: Test image 1

After uploading the image, we had to click the predict button and after clicking the button, our page showed the output for this image as Figure 5.2:

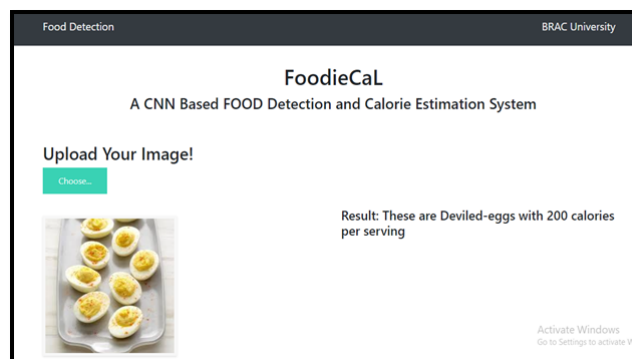


Figure 5.2: Output for test image 1

From this input image, we can see that these are deviled eggs and our Single Food Detector System was already trained with around 1000 images of deviled eggs. So,

our system was successful in detecting the food item and it also printed the calories for deviled eggs with 100% confidence. We also got a prediction chart for all the 20 foods in our dataset as followed:

<b>Food Category</b>	<b>Prediction Percentage</b>
Chicken Wings	79.86%
Chocolate Cake	22.16%
Churros	33.14%
Cup Cakes	92.02%
<b>Deviled Eggs</b>	<b>99.97%</b>
Donuts	56.90%
French Fries	31.53%
Fried Rice	34.78%
Grilled Cheese Sandwich	52.91%
Hamburger	45.96%
Hot and Sour Soup	30.94%
Hot Dog	55.03%
Macarons	53.86%
Oysters	62.63%
Pancakes	87.87%
Pizza	24.59%
Samossa	45.63%
Seaweed Salad	4.3%
Steak	19.99%
Waffles	82.48%

Table 5.1: Prediction percentage for test image 1

From the image of Deviled eggs, our system predicted that:

The probability of the image being Chicken wings is 79.86%

The probability of Chocolate Cake is 22.16%.

The probability of Churros is 33.14%.

And our system predicted that the probability of the image being Deviled eggs is 99.98%.

The table 5.1 is showing the probability of all our food items after detecting the input image and it is only printing the outputs where prediction percentage is above 96%. To be more specific, we kept the threshold of our prediction percentage at 96 and any food item below that accuracy will not be shown as output.

Then we proceeded to test our system with more images. We uploaded another image and clicked the predict button and our system showed the output like shown in Figure 5.3:

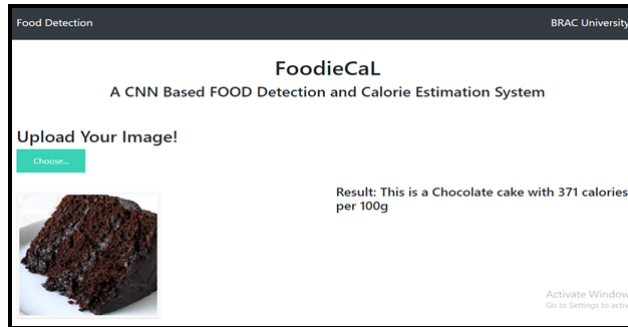


Figure 5.3: Output for test image 2

As we can see from this image, this is a chocolate cake. As our system was trained with 1000 images of chocolate cake, our system was able to detect it from the given image. It also printed the calories per slice in a chocolate cake and printed the accuracy as well. We also got a probability chart for all the 20 food categories in our dataset as followed:

Food Category	Prediction Percentage
Chicken Wings	44.26%
<b>Chocolate Cake</b>	<b>99.99%</b>
Churros	57.65%
Cup Cakes	92.21%
Deviled Eggs	47.90%
Donuts	83.82%
French Fries	33.86%
Fried Rice	65.59%
Grilled Cheese Sandwich	69.71%
Hamburger	12.98%
Hot and Sour Soup	13.02%
Hotdog	14.81%
Macarons	45.40%
Oysters	17.45%
Pancakes	71.89%
Pizza	45.86%
Samosa	21.81%
Seaweed Salad	40.38%
Steak	85.08%
Waffles	58.16%

Table 5.2: Prediction percentage for test image 2

From the image of Chocolate cake, our system predicted that:  
 The probability of the image being Chicken wings is 44.26%  
 The probability of Deviled eggs is 47.90%  
 The probability of Churros is 57.66%

And our system predicted that the probability of the image being Deviled eggs is 99.99%.

The Table 5.2 is showing the probability of all our food items after detecting the input image and it is only printing the outputs where prediction percentage is above 96%. To be more specific, we kept the threshold of our prediction percentage at 96 and any food item below that accuracy will not be shown as output. Here only Chocolate cake is detected because only chocolate cake has prediction percentage above 96%.

## 5.2 Analysis on Multiple Food Detection

After designing our webpage, we started to test out system with some sample images. We chose an image using the Choose button and uploaded an image in our website (Figure 5.4). After uploading, our webpage looked like this:

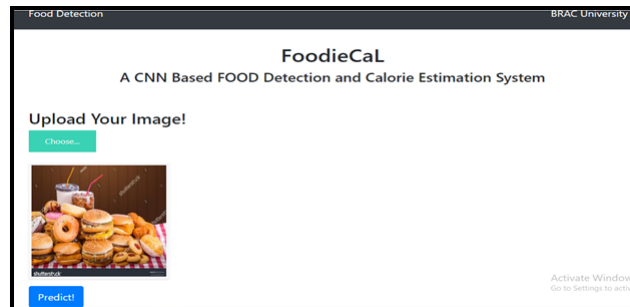


Figure 5.4: Test image 3

After uploading this image, we clicked the Predict button and the output was shown like the image given below in Figure 5.5:

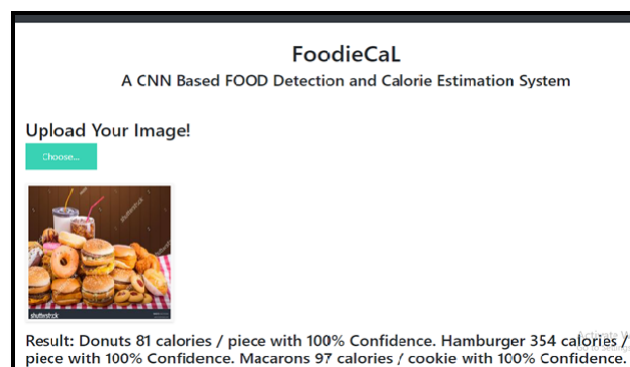


Figure 5.5: Output for test image 3

This test image contained Hamburgers, Donuts, Macarons and Milkshakes. We had hamburgers, donuts and macarons in our dataset and our CNN was trained to predict these items from an image. As we did not have milkshakes in our dataset, our system did not detect milkshake. As we can see from the output, our system successfully predicted that the image contained hamburgers, donuts and macarons



as well. Our system also printed the calories for these food items and also showed the prediction accuracy as output. We also got a probability chart for this image for all the 23 categories from our dataset. The probability chart was shown in the command prompt as followed:

<b>Food Category</b>	<b>Prediction Percentage</b>
Chicken Wings	25.10%
Chocolate Cake	24.65%
Churros	89.12%
Cup Cakes	88.44%
Deviled Eggs	54.64%
<b>Donuts</b>	<b>99.89%</b>
Fried Rice	25.06%
Grilled Cheese Sandwich	92.29%
<b>Hamburger</b>	<b>99.69%</b>
Hot and Sour Soup	09.47%
Hotdog	93.66%
Ice Cream	56.85%
Lobster Bisque	41.54%
<b>Macarons</b>	<b>99.73%</b>
Oysters	28.03%
Pancakes	95.17%
Pizza	07.99%
Samosa	23.62%
Seaweed Salad	01.89%
Steak	11.76%
Sushi	78.32%
Tacos	10.68%
Tuna Tartare	30.79%
Waffles	82.20%

Table 5.3: Prediction percentage for test image 3

From the image shown in Figure: 5.5, our system predicted that:

The probability of the image being Chicken wings is 25.10%

The probability of Donuts is 99.88%

The probability of Hamburger is 99.69%

And our system predicted that the probability of the image being Macarons is 99.72%.

The table 5.3 is showing the probability of all our food items after detecting the input image and it is only printing the outputs where prediction percentage is above 96%. To be more specific, we kept the threshold of our prediction percentage at 96 and any food item below that accuracy will not be shown as output. Here Hamburger, Donuts and Macarons are detected because their prediction percentage are above 96%.

Again, we tested our system with new images. We chose an image from our local

directory and uploaded the image in our webpage. After uploading, we clicked the predict button and then we got this output as given below in Figure 5.6:

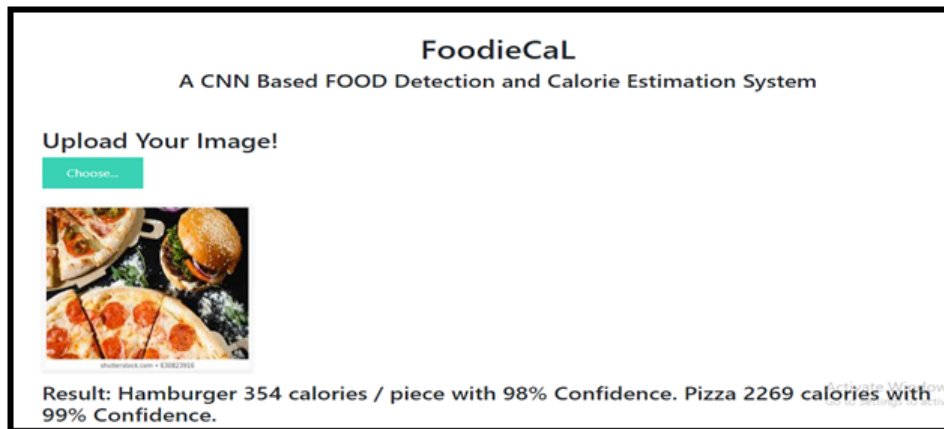


Figure 5.6: Output for test image 4

From this test image, we can see that there are 2 pizzas and a hamburger. Our system was trained with 1000 images of hamburger and also trained with 1000 images of pizza. As a result, our system was able to detect both of these food items successfully. Our system is saying that it is 98% certain that there is a hamburger in this image and it is 99% confident that there are pizzas in this image as well. We also got a probability chart for this image for all the 23 categories from our dataset.

From this image, our system predicted that:

The probability of the image being Chicken wings is 27.10%

The probability of Fried Rice is 59.50%

The probability of Hamburger is 98.26%

And our system predicted that the probability of the image being Pizza is 99.16%.

The table 5.4 is showing the probability of all our food items after detecting the input image and it is only printing the outputs where prediction percentage is above 96%. To be more specific, we kept the threshold of our prediction percentage at 96 and any food item below that accuracy will not be shown as output. Here Hamburgers and Pizza are detected because their prediction percentages are above 96%. The probability chart was shown in the command prompt as followed:

<b>Food Category</b>	<b>Prediction Percentage</b>
Chicken Wings	27.10%
Chocolate Cake	20.37%
Churros	19.72%
Cup Cakes	63.96%
Deviled Eggs	78.12%
Donuts	49.81%
Fried Rice	59.50%
Grilled Cheese Sandwich	53.69%
<b>Hamburger</b>	<b>98.26%</b>
Hot and Sour Soup	28.49%
Hotdog	65.89%
Ice Cream	25.05%
Lobster Bisque	21.19%
Macarons	54.03%
Oysters	88.23%
Pancakes	79.65%
<b>Pizza</b>	<b>99.17%</b>
Samosa	35.62%
Seaweed Salad	09.45%
Steak	67.05%
Sushi	85.84%
Tacos	59.89%
Tuna Tartare	85.08%
Waffles	58.16%

Table 5.4: Prediction percentage for test image 4

### Analyzing Top Food Items:

For this section, we randomly chose 5 food items and 4 test images shown in Figure 5.7 which have these selected food items and we analyzed the prediction accuracy for these 5 food classes. We got prediction percentages for all the 23 food classes in our command prompt. We only took the prediction percentages for these 5 food items and showed the percentages in a tabular form. In the table 5.5, we have shown all the percentages respective to the 4 images uploaded in our webpage as input:

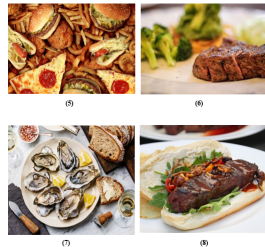


Figure 5.7: More Test images

Test Image No.	Pizza	Hamburger	Hotdog	Steak	Oysters
5	97.37%	98.02%	89.24%	44.69%	86.08%
6	71.75%	62.25%	19.30%	99.98%	49.06%
7	50.77%	88.25%	24.02%	48.80%	99.88%
8	58.94%	97.74%	99.72%	97.89%	36.18%

Table 5.5: Output comparison for some test images

### 5.3 Limitations

In all of the papers we read on the food detection, they had some limitations in their system. Some of them had low accuracy, some of them could not detect multiple foods accurately and so on. In this paper, we tried different models to overcome these limitations and we managed to overcome some of these limitations as we have shown earlier with our result analysis. However, we found out that our Multiple Food Detecting System has some flaws too. These flaws are not severe but as they are the limitations of our system and as a result, we have to show them. All of our limitations occurred for the inconsistency in the images in our Dataset. For example:

- Hot and Sour Soup and Lobster Bisque in our Dataset are pretty similar. So, when we upload some test images of Lobster Bisque in our system, it was predicting that the image is of both Lobster Bisque and Hot and Sour Soup.
- Another example is: Hot Dogs and Tacos are similar looking and when we input some images of Tacos, our system predicts that the image can be both Tacos and Hot Dog. Tacos will have better accuracy but as we kept the threshold at 96, Hot Dog is also shown in the output.
- Lastly, the angle at which the image is taken is very important for our system to detect food items accurately. For example: all of the images in the Hamburger dataset are taken from one side so that the patty is visible. So, if we input an image of Hamburger from the top view, the patty will not be seen by our system and so our system will not be able to detect Hamburger in that scenario.

Finally, we are certain that if we work with better and bigger dataset, we can surely overcome these limitations.

# Chapter 6

## Conclusion and Future Work

Our main purpose of this study is to identify the calorie of the food from a given image which will help people overcome diseases like obesity, diabetes, heart problem, kidney failure, high blood pressure and other diseases caused by being overweight. We believe that if people know about the calories of the food, it will help them to lead a healthier life by keeping track of how much calories they are consuming. We have researched and also looked into various methods for the food recognition process. We have studied deep learning and machine learning and feature extraction techniques such as ResNet, MobileNet, Inception etc. We have also studied Max-Pooling and some api such as keras, tensorflow etc.

CNN is most suitable for image or object detection processes. CNN can provide or give better outputs than other machine learning and deep learning algorithms. In case of performance, CNN outperforms other neural networks and machine learning and deep learning algorithms on conventional 2D or 3D image recognition tasks and other object detection tasks. CNN's are also useful for single dimension problems like time series, and in our case, 3D image classification where the images we used are food. In terms of statistical results, they also have high calculating efficiency in terms of object or image classification system.

At this stage of our research, we focused on building a Webpage based on CNN algorithm which is able to classify different types of foods and show us the nutrition value of these foods. In the upcoming stages, we are planning on developing a mobile app that not only identifies the food items with a great accuracy from an image captured on the smartphone but also it will review the medical reports of the user to suggest whether the amount of calorie should be taken or not. Moreover, in the near future we want to work with a bigger dataset for example: a dataset of 100 or more Bengali food items using this model we have shown in this paper. As such a dataset is not available, we are thinking of making our own dataset with 2000 or more images for each food category.

We think this app will be very beneficial for this generation because people are obsessed with junk foods these days and they also do not want to get obese. So, while eating these high calories contained foods, they just have to take a picture using our app and can keep track of how many calories they are consuming in that particular time.

# Bibliography

- [1] T. Joutou and K. Yanai, “A food image recognition system with multiple kernel learning,” in *2009 16th IEEE International Conference on Image Processing (ICIP)*, IEEE, 2009, pp. 285–288.
- [2] H. Hoashi, T. Joutou, and K. Yanai, “Image recognition of 85 food categories by feature fusion,” in *2010 IEEE International Symposium on Multimedia*, IEEE, 2010, pp. 296–301.
- [3] M.-Y. Chen, Y.-H. Yang, C.-J. Ho, S.-H. Wang, S.-M. Liu, E. Chang, C.-H. Yeh, and M. Ouhyoung, “Automatic chinese food identification and quantity estimation,” in *SIGGRAPH Asia 2012 Technical Briefs*, 2012, pp. 1–4.
- [4] Y. Kawano and K. Yanai, “Real-time mobile food recognition system,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 1–7.
- [5] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [6] V. Bettadapura, E. Thomaz, A. Parnami, G. D. Abowd, and I. Essa, “Leveraging context to support automated food recognition in restaurants,” in *2015 IEEE Winter Conference on Applications of Computer Vision*, IEEE, 2015, pp. 580–587.
- [7] E. J. Gallagher and D. LeRoith, “Obesity and diabetes: The increased risk of cancer and cancer-related mortality,” *Physiological reviews*, vol. 95, no. 3, pp. 727–748, 2015.
- [8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.
- [9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [10] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.
- [11] N. C. Institute, *Obesity and cancer risk*, <https://www.cancer.gov/about-cancer/causes-prevention/risk/obesity/obesity-fact-sheet>, 2017.
- [12] T. Gupta, *Deep learning: Feedforward neural network*, Dec. 2018. [Online]. Available: <https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7>.

- [13] N. Martinel, G. L. Foresti, and C. Micheloni, “Wide-slice residual networks for food recognition,” in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2018, pp. 567–576.
- [14] S. Saha, *A comprehensive guide to convolutional neural networks-the eli5 way*, Dec. 2018. [Online]. Available: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.
- [15] M. A. Subhi and S. M. Ali, “A deep convolutional neural network for food detection and recognition,” in *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, IEEE, 2018, pp. 284–287.
- [16] R. Balsys, *Convolutional neural networks (cnn) explained step by step*, Feb. 2020. [Online]. Available: <https://medium.com/analytics-vidhya/convolutional-neural-networks-cnn-explained-step-by-step-69137a54e5e7>.
- [17] *Feedforward neural network*, Dec. 2020. [Online]. Available: [https://en.wikipedia.org/wiki/Feedforward\\_neural\\_network](https://en.wikipedia.org/wiki/Feedforward_neural_network).
- [18] V. Kurama, *A guide to resnet, inception v3, and squeezenet*, Jun. 2020. [Online]. Available: <https://blog.paperspace.com/popular-deep-learning-architectures-resnet-inceptionv3-squeezenet/>.
- [19] W. H. Organization, *Obesity and overweight*, <https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight>, 2020.
- [20] Shashikant, *Convolutional neural network: A step by step guide*, Jan. 2020. [Online]. Available: <https://towardsdatascience.com/convolutional-neural-network-a-step-by-step-guide-a8b4c88d6943>.
- [21] *Back propagation neural network: Explained with simple example*. [Online]. Available: <https://www.guru99.com/backpropogation-neural-network.html>.
- [22] *Using the keras flatten operation in cnn models with code examples*. [Online]. Available: <https://missinglink.ai/guides/keras/using-keras-flatten-operation-cnn-models-code-examples/>.