Electronics and Computer Science

Faculty of Engineering and Physical Sciences

University of Southampton

Palak Prakash Jain

23/09/2021

A Comparative Study of Food Detection and Classification Techniques

Project supervisor: Dr. Jonathan Hare

Second examiner: Sasan Mahmoodi

Project final report submitted for the award of

MEng Computer Science

Word count: 9957

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

ELECTRONICS AND COMPUTER SCIENCE

A progress report submitted for the award of

MEng Computer Science

by Palak Prakash Jain

According to the World Health organization, more than 1.9 billion adults in 2014 have been classified to be in the overweight category. Alternatively, nearly 39% of the world's adult population is overweight or 13% is obese[126]. These are alarming figures and there is a need more than ever to control and record eating habits. In fact, over the past 40 years, obesity figures have almost doubled[126]. Mobile applications are increasingly being used both to share images of food and to record food intake of individuals, and the number of images of food on the internet is growing exponentially every day. As a result, demand for automated and reliable classification of food is at its peak.

The purpose of this project is to compare the various robust methods of object detection and classification and apply them to the domain of food recognition, by exploring deep neural networks and traditional approaches. Experimentation with various neural networks and traditional approaches has been carried out on openly available food datasets to gain a deeper understanding of which are the best suited approaches for this task. The goal of the final work is to adjust and train the models optimally to the chosen datasets, in order to obtain highest accuracy on unseen data.

STATEMENT OF ORIGINALITY

I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.

I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.

I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

1. I have acknowledged all sources, and identified any content taken from elsewhere.
2. I have not used any resources produced by anyone else.
3. I did all the work myself, or with my allocated group, and have not helped anyone else.
4. The material in the report is genuine, and I have included all my data/code/designs.
5. I have not submitted any part of this work for another assessment.
6. My work did not involve human participants, their cells or data, or animals.

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Description

The food we eat plays a prominent role in influencing our lives. Multiple factors affect our food choices, including our taste, ethnic, social and family background, and health and dietary requirements[36]. Recent changes in mobile phone technology, development of high quality camera phones, tables and advanced software in electronic devices have powered the use of emerging food detection technology[43]. The manual process of food identification is slowly becoming obsolete.

An example of highly-regarded work in the area of Food Detection is an application released by Team Clarifi, who emphasize that their model has the ability to recognize hundreds of different types of foods, including their ingredients composition. The model was apparently built with a number of different applications involving recognition, retrieval or recommendation, in mind. The latest application of Food Detection can range from the field of travel, medical services, or general lifestyle, to physical health and hospitality industries centered around food. E-commercial websites, restaurants, customer brands and platforms can also make use of Food Recognition to analyse users' reviews automatically to improve their products/services[10].

Modern technologies include electronic noses, computer vision, spectroscopy and spectral imaging etc are quite highly capable of recognising the characteristics of the food. This process can be made very fast and automatic, where a program analyses the food and informs us about its components or nutritional features [2]. Computer Vision has two principal methods: Traditional and Deep Learning. The DL method has overcome the majority of issues of Traditional methods which resulted in poor recognition accuracy and classification of its constituent ingredients etc. but yet, has its own limitations[30].

1.2 Project Description

This research paper is a comparison study between the performance of recent Deep Learning models and Traditional approaches in this domain. The Deep Learning models were originally trained on the ImageNet dataset and then fine-tuned to Food datasets such as Food-11[69] and Food-101[15]. Models will be optimised throughout the development and comparison to maximise performance.

Chapter 2

Background Research and Literature Review

2.1 Traditional techniques

Traditional approaches in Object Detection in general follow a certain pattern. Firstly, they use the region selector to attempt to find the object's location. This scans through the image, by a fixed factor each time, and uses sliding windows of various sizes to generate image blocks of equal size. Next, the key visual features of the image patches are extracted when the feature extractor utilises techniques like SIFT, SURF, FAST, Hough Transforms and Geometric Hashing. Finally, the distinct object category is classified with the help of Support Vector Machine or Adaboost [47].

The most popular algorithm techniques are as follows:

- SIFT: Most widely used technique; it has the capacity to identify large numbers of features from images. It is also used for description of features [70].
- SURF: To some extent it is similar to SIFT but it is faster due to less computation time as the core focus is on integral images for convolutions [70].
- FAST: As the name suggests, it is used in the areas where time required is very less like processing real time video. It does not have a feature descriptor, and instead of feature detection it works with the concept of corner detection [70].
- Hough Transforms: This is mostly used in the case of parametric curves , circles lines or similar geometric images, particularly when circle detection is to be done on a real time basis [71].
- Geometric Hashing: This approach works with an indexed database of features of geometric objects, it recognises the object very fast, and it is not necessary that the database should have already defined features for matching images [72].

SIFT Detector

The SIFT detector is used to identify objects which have very specific or distinct features and these features do not vary with scale, rotation, illumination, noise or blur. It works with identification of keypoints, stable locations, the scale and features of the object and, most importantly, uses a descriptor to match an image to another.

The key steps are[82]:

- Creating a scale space using the convolution of the Gaussian Blur operator.
- Keypoint finding and localization.
- Ensuring keypoints do not change with different orientation assignments.
- Use of descriptors to describe keypoints.
- Matching of the keypoints.

The process starts with the convolution of the Gaussian Blur operator and the image. The blur of an image is represented by applying the expression: "$G(x, y, \sigma) = \frac{1}{2\Pi\delta^2} e^{-(x^2+y^2)/2\sigma^2}$" on every pixel of the original image[80]. The Gaussian Blur Operator (G), pixel location (x,y) are coordinates; the parameter σ adjusts the scale of the function. Function L(x,y,σ) with the formula, " L(x,y,σ) = G(x,y,σ) * I(x,y)" (Koenderink (1984) and Lindeberg (1994)) are used to calculate the scale-space. A different value function G is used each time to create an image of variant blur [55].

The difference of the Gaussian Blurring of an image with two different σs, σ and (k*σ). It is calculated using the formula: "D(x, y, σ) = L(x, y, kσ) - L(x, y, σ)" (Lowe 1999). The function is repeated for every octave. The results of the difference of Gaussian methods applied in each octave is used to find out the scale-invariant Gaussian approximations. Identification of key points is then achieved by changing the orientation and localisation parameters to remove low contrast points and edges. An image histogram is used to describe the gradients around keypoints to identify distorted shapes [55].

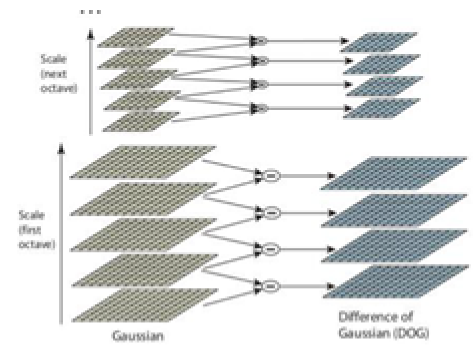


Fig. 1: Difference of Gaussian method (Sourced from: [80])

Fig.1 depicts two different octaves of the scale space pyramid and image scales in ascending order in each octave. The difference between two consecutive scales is calculated. Two successive images belonging to an octave are selected and one is subtracted from the other. The following set of two is then taken, and the process repeats for all octaves. The final images are very beneficial for identifying key points. We then calculate the Laplacian of Gaussian approximations that tell us which keypoints do not vary with scale[80].

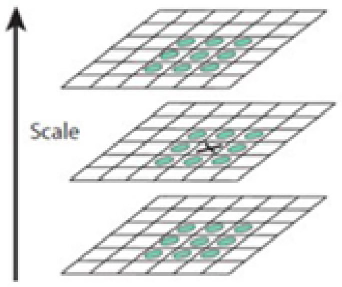


Fig. 2: Comparison of keypoint with neighbours(scale invariant) (Sourced from: [80])

In the first step we calculate the difference of Gaussians followed by Laplacian of Gaussian approximations which tells us keypoint that do not vary with scale [80]. Consider Fig. 2; The x keypoint represents a potential keypoint which is surrounded and compared to the immediate 8 neighbours as well as the 9 pixels in the next scale and in the previous scale. In case it's a "local extreme point"(either minimum or maximum in value), then it is a potential keypoint which means this keypoint is best represented in that scale. Mathematical Taylor series expansion of scale space is typically used to get a more accurate location of extrema. And removing extrema which falls to a value less than the 0.03 threshold[80].

Advantages of SIFT[55]:

- Features identified by SIFT are notably distinct, making them relatively straightforward to find their right matches within a large database.
- Very efficient and near to real-time performance.
- Can be expanded effortlessly to a variety of different types of features, with greater robustness.
- Higher accuracy rates
- SIFT does not vary key points with image scale and rotation, this makes it the most popular approach
- Available with open CV library, easy to access and use
- Image noise is filtered very efficiently in SIFT

Drawbacks of SIFT[55]:

- It requires high level of mathematical and computational understanding and aptitude
- It is time consuming because the gradient of each pixel needs to be computed

Other feature detectors include:

PCA-SIFT Detector

Principal Component Analysis is a method typically used to reduce dimensions and transform a large database of features to a smaller one though with a bit of compromise on accuracy. The core principle is to find common features between images based on the calculated distance between both image vectors. The result is fewer components, meaning less storage space is used up. Thus, it leads to faster matching and therefore less time is taken in computation[37][84].

SURF Detector

Unlike the SIFT detector, the SURF detector builds a ‘stack’ with the same resolution as the image. It also does not vary with scale, rotation , blurring etc. The key advantage of SURF is its speed as it uses the box filter and integral images for image transformation. In the situation of rotational and blurring, this approach is more stable than SIFT[37][85][86].

Bag of Visual Words

Bag of Visual Words is based on the concept of identifying distinct image features similar to classifying words in the Bag of Words concept, where language (rather than images) is used for information processing. Here, the key feature points of the image are identified which remain unaltered in any situations such as rotation or scaling. The links between patterns in different images assist us in matching and identifying their image category. There is a descriptor which identifies the keypoint and forms a cluster around it, using information from other keypoint descriptors. A frequency histogram stores the information that results from the clustering process, for the process of identifying similar images. This histogram is referred to as the ‘Bag of Visual Words’[37][87].

Traditional algorithms discussed in this section are very simple compared to Deep Learning and perform more efficiently with less programming required. These algorithms are independent of the image class and are common in nature and work the same for any image feature. This is significantly different from the key fundamentals of a Deep Neural Net. where the preparation of the training data set is very important. Preparing a large data set requires time, resources and research. This long process may prove to be inefficient at times. Consider an example of a production process, where two balls of separate colours need to be classified. In this case, simple colour thresholding will be more useful than creating a Deep Learning Model. Significant improvement in processing power, however, has led researchers to be more inclined towards a data-driven approach in image recognition. Over the past decade specifically, researchers have reached accuracy rates comparable to human vision with data-driven approaches on tasks based on neural networks, and they have outperformed the Traditional techniques of solving Computer Vision problems[47][88][89].

2.2 Deep Learning Approaches

The field of Computer Vision has evolved over the years from machines working for humans to today's world where machines are mostly independent and quasi human. Deep learning has changed the whole perspective. The machine capabilities have improved over the years with better computing speed, memory, improvement in image sensing and optic technology, cost and power efficiency. With the advent of Deep Learning, difficult problems like Image modification or colorization, segmentation, and better prediction are now possible. Compared to the traditional algorithm of CV, Deep Learning models are orderly but complex [90]. In short we can compare the Traditional techniques with a small car used by families for local shopping and going to work or school, whereas Deep learning can be compared with a big SUV which runs on highways. Most commonly used Deep Learning models for image classification are Convolutional Neural Networks (CNNs) which typically improve performance of Image recognition significantly with large databases and computing resources that have achieved superhuman accuracy in Image classification[47].

2.3 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a Deep Learning algorithm which primarily involves “processing, classification, segmentation, and categorisation of features in images” in the field of Computer Vision. [91] The algorithm is based on convolution which is a linear operation in which two inputs are multiplied and the outcome in turn is used to extract the features of the image.This process is essentially ‘helped by’ multiple layers while processing pixel data. The whole process is as follows: First the key features of the image are extracted, then the algorithm runs through different layers like convolution, pooled and connected layers, and finally a vector of probabilities is given as output to identify the image[91](Illustration of CNN in Fig. 3).

Basic Convolutional Neural Network Architecture

The CNN architecture is built from the understanding of how the neurons are connected and how the visual cortex is organized and functions, influencing and patterning the way neurons work in a human's brain[61]. For example when we see a monkey, we see the eyes, nose, face and go on and then we recognise it is a monkey, similarly CNN identifies the image in a similar way. CNN arranges the neurons to process visual inputs like the human brain. The order and set-up of these layers is the foundation which forms the architecture of CNN [92] [93] [94].

- **Convolutional layer**: This is the first layer which receives images as input and extracts /identifies the features with the help of the feature map based on a mathematical convolution calculation while scanning through the entire image. This is then utilized in the prediction of different class probabilities for the features.Convolutional layers sum up the presence of specific information/attributes in the input image. Furthermore, the deeper layers discover more abstract or higher-order features, such as particular shapes or objects. The features located in the input can influence the output. [6][94].
- **Pooling layer** : The information received from several feature maps are pooled together to arrive at the most precise relevant features. It basically scales down the convoluted feature map and only keeps the most important information. This is achieved by reducing the existing connections that link the layers and individually functioning on every feature map. (The convolutional and pooling layers usually repeat for a number of times.). With the pooling layer, the accuracy decreases, but it minimises the required calculation and keeps the end result safe, resulting in a more efficient network . Consider an example where a pooling operation of size 5×5 pixels is applied to an image with a stride of 5 pixels. This means that the pooling layer reduces each feature map size by a factor of 5. This can be explained with an example. Suppose we want to identify a lion. It is not necessary to identify each component of its face precisely like the eyes, nose, and the teeth, as we know all these are near to his head or between [128].

  The 3 main functions used in the Pooling operation are: [6]

- ❖ **Average Pooling**: This takes the average of elements from the pixel block of the feature map so that output has some attributes of less important elements of the feature map.
- ❖ **Maximum Pooling** (or Max Pooling): The largest or maximum element is selected from the pixel block of the feature map so that output has maximum attributes of the feature map. This helps in reducing the size.
- ❖ **Sum Pooling** : The sum of all the values in each patch is calculated [94] [95].

- **Fully connected layer** (Dense layer): Weights are assigned to the output achieved by feature analysis to predict the correct class for each image fed in as input.

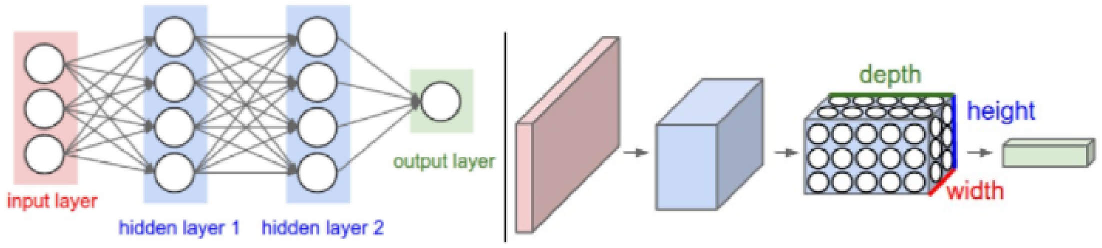
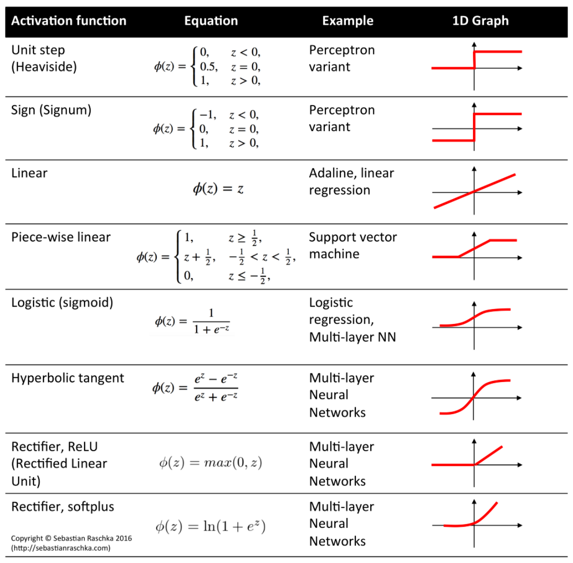


Fig. 3: Left: A pictorial side-by-side comparison of a typical layered Neural Network vs a Convolution Neural Network which arranges its neurons into 3 dimensions (depth, width and height), specifically shown in the third layer. (Sourced from: [123]).

As we have seen during our explanation of a Neural Network, linear layers have certain limitations. This requires some review of the Non linear or Dense layers, which are based on the concept of activation and work with the same formula like Linear layers “wx+b”. but the final result is calculated using the nonlinear function: $y = f(wx + b)$. The layers bring non-linearity in the network and help in deciding attributes to be carried till the end or to be dropped; this means the model learns parameters w and b with f either being a linear or a non-linear activation function[41].



Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

Copyright © Sebastian Raschka 2016 (http://sebastianraschka.com)

Fig. 4: List of activation functions (Sourced from: Sebastian Raschka [96])

Mathematical Taylor series of functions for infinite sums can be used to decompose the activation function. The polynomials of higher degrees than 1 can be created with non-linear layers.“Consider using the polynomial activation function: ($y=x^2+x$). By layering two instances of this, a polynomial of degree 4 containing the terms (x^4, x^3, x^2, x) can be produced.” [96] In this way, we can model the network using more layers and more complex mathematical functions. Activation functions can totally change the weight and activation of the input and subsequent output [41] Fig. 4 sums up some of the most-used activation functions in Neural Networks.

ReLU

The Rectified Linear Activation Function(ReLU) has been used in the project, as it is the standard function used in several varieties of neural networks. ReLU is a piecewise linear function that outputs the input directly if it is greater than 0, or else outputs zero [42] (as shown in Fig. 5 below). The model that utilizes this ReLU is usually simpler to train and performs better. ReLU works with a model like ReLu(x)=Max (0,x) [94] [97] [98].

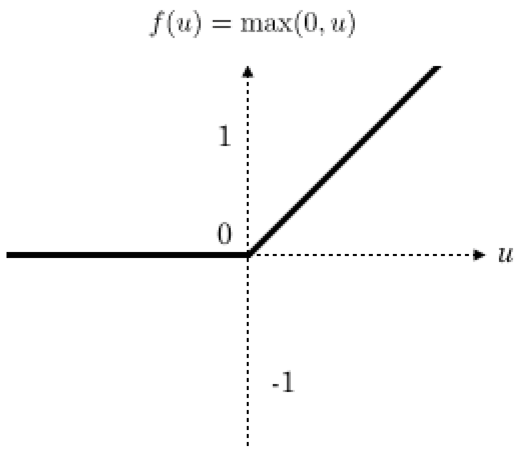


Fig. 5: RELu Function; (Sourced from: [124])

Advantages of ReLU :

- Simple
- No heavy computation unlike sigmoid
- Less time to train or run
- Sparsity
- Avoids vanishing gradient problem

Issues

- Reverse of vanishing gradient leads to piling up of gradient errors, restricting model flexibility and stability.
- The negative side of the graph may be fixed and result zero , resulting in a large part of the network not working.[43].

The Dropout Layer

During the process of layering, ultimately all the features get connected to a Fully Connected layer. At this layer, the FC layer adopts these features, which results in an increase in parameters. This increase in parameters leads to overfitting of the training data set and impacts the overall performance of the model as it becomes very complex. The Dropout layer is introduced to mitigate the impact of overfitting and bringing back the optimal performance of the data set. In this process some neurons are taken out from the network in the training process to make the model lighter[5]. Dropping of nodes may create issues temporarily but left out nodes have to take care of the quality of input key characteristics and accuracy. In this stage the mistakes from earlier stage also get corrected which makes the model better. This Ensemble model with larger computations has a few limitations in dealing with several large models as more time is required, which may not be easy [101] [102].

There are a number of advantages of deploying Dropout Regularization.

- Can be used with all Network Types

- It takes care of the problem of overfitting in large networks. Network weights will increase in size on removal of layer activations. The incidence of error is reduced in large datasets due to Dropout regularisation. Though it increases the cost a bit, it requires cost-benefit analysis. generally it is being experimented to arrive at the ideal "drop out rate of 0.5 to 0.8" and "the test is being done gradually from 0.1 to 1" [103] .

2.4 Transfer Learning

In Machine Learning, Transfer Learning plays a pivotal role. A model prepared earlier for a particular task serves as a standard model which is used with or without modification for some other job as a first step, the jobs involved here are mostly similar or related in the computer vision field. This saves time and effective utilisation of resources as with little or no data can serve the purpose [6]. There are two frequently used techniques in Transfer Learning: The Develop Model and The Pre-trained Model methods. Fig. 6 depicts the steps taken through the Develop Model Approach.

Develop Model Approach[54]

1. Selection of Model task: A related or similar task is selected with a big database having relationship in the input/output data, alongwith concepts or clear path from input to output data mapping. The model task outcome can be predicted and can be related or similar.
2. Development of the Source Model: Develop a reasonable model with clear logic for the initial task. This is relatively much more efficient than developing a fresh model each time.
3. Reuse Model: In this case any model which works perfectly with the current source task is being modified as the starting stage for a model on the second task. Different parts of the Source model based on techniques used in modelling can be used.
4. Tune Model: In another possibility which depends on the job, the current model can be readjusted or modified based on the existing input-output database as per requirement.

Pre-trained Model Approach

Any existing model chosen from the list of robust models can be used to start with as per requirement of a task, these can be modified on input or output based on task requirements. The key difference here is the absence of the Develop Model stage in this approach, assuming the model is already trained on the initial large dataset(Usually ImageNet).

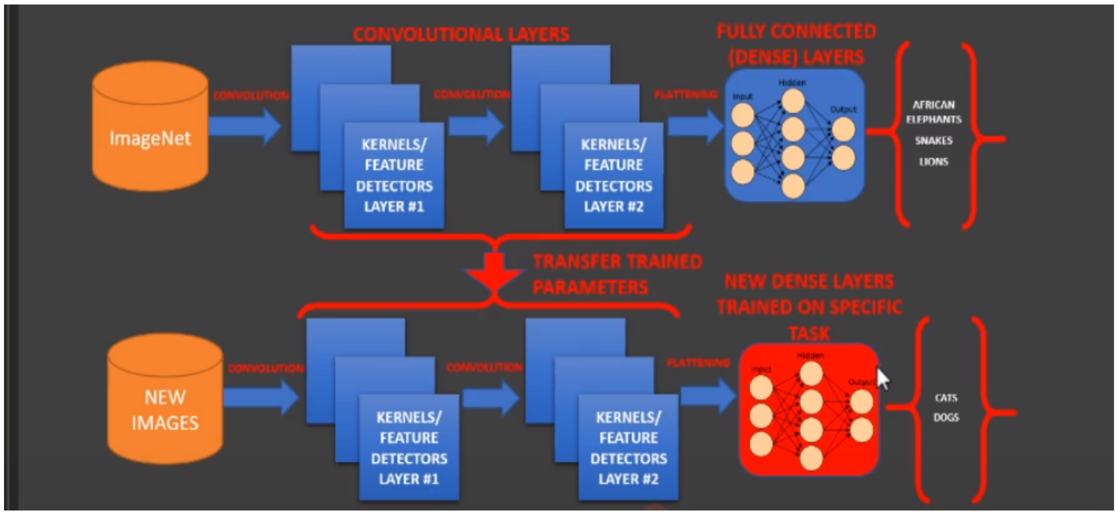

CONVOLUTIONAL LAYERS
FULLY CONNECTED (DENSE) LAYERS
ImageNet
CONVOLUTION
KERNELS/ FEATURE DETECTORS LAYER #1
KERNELS/ FEATURE DETECTORS LAYER #2
FLATTENING
Input
Hidden
Output
AFRICAN ELEPHANTS
SNAKES
LIONS
TRANSFER TRAINED PARAMETERS
NEW DENSE LAYERS TRAINED ON SPECIFIC TASK
NEW IMAGES
CATS
DOGS

Fig. 6: Transfer Learning: Pre-trained Model Approach (Sourced from: [125])

Popular Convolutional Neural Network Architectures

The design of the architecture of a CNN, structure of layers, the elements underneath are all key factors which are responsible for its speed, efficiency, reliability, accuracy and performance of tasks.

ImageNet is a large visual revolutionary database containing millions of images, some of which also contain bounding boxes for networks such as YOLO, designed specifically for utilization in the field of object recognition. In the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC), competitors create models that attempt to accurately detect and classify objects and/or scenes in the images they are trained[26]. Classification rates are improving year on year from 26% to AlexNet 15.3% [62].Recently , the error rates have fallen further, even exceeding human capabilities [104].

The differences in architecture and their advantages:

Several general CNN architectures exist, most of which have received recognition by achieving good results at the ILSVRC. The ILSVRC makes use of 1000 classes with 1000 images each in the Imagenet database. There are nearly 1.2 million training, 50,000 validation and 100,000 testing images. Consider the architecture of LeNet-5 (1998); it comprises 7-layer and 32×32 pixel grayscale input images which are digitised and were used by mostly financial institutions to digitise checks made manually [42].

- AlexNet (2012)

Alex net has five convolutional layers, 3 connected layers along with stacked Convolutional layers. With the support of multiple GPUs i.e two Nvidia GTX 580 GPUS, it can split its network to two segments, the advent of overlapping reduces the error. It had an overfitting problem due to large databases and parameters, but with the help of Dropout and Data Augmentation this overfitting issue was contained [42].

- GoogleNet (2014)

GoogleNet has 22 layers, almost 3 times the number of layers as AlexNet, and it is very accurate, almost the same as human sight with an error rate of only 6.67%. Though it is inspired by LetNet, it has been improvised to include the Inception module. Another key characteristic of the GoogleNet architecture is the reduction of parameters by millions compared to AlexNet of only 4 million as it works with small convolutions and batch normalisation [42] [105] [106].

- VGGNet (2014)

VGGNet has 16 convolutional layers and a uniform architecture. It works with 3×3 convolutions along with several additional filters. The main issue with VGG is the abundance of parameters(almost 25* more than GoogleNet, and 3* more than AlexNet), but its training on 4GPUS enhances its performance [42] [107] [47].

2.5 Existing Food Detection and Classification Implementations

In 2009, a researcher built the PFID composed of 4545 images of 101 different food classes. An SVM classifier was applied to this dataset resulting in an accuracy of 11% and 24 % respectively for classification on the Color histogram method and the Bag of SIFT feature extraction method [56].

In 2009, a researcher built the PFID composed of 4545 images of 101 different food classes. The SVM classifier was applied on this dataset and it achieved a classification accuracy of 11%, with the color histogram method and 24 % with the bag-of-SIFT-features method[56]. Later on, Rashed Mustafa and Prashenjit Dhar proposed a method with combined Gist and SURF features, which along with the SVM classifier results in 93.3% accuracy, compared to 84.4%, 80.5% and 68.7% using Adaboost, Decision Trees and KNN models respectively, highest recorded using traditional approaches[53].

After achieving almost perfect accuracy results with the binary classification of food, research has progressed into multiclass food classification. Several deep learning-based classifiers were trained on the Food-101 dataset. The most used indicators used are Top-1 and Top-5 classification accuracy. In 2015 Yanai and Kawano with a fine-tuned version of AlexNet got 70.41% top-1 accuracy[34]. Liu et al. with the Deep Food network reached 77.40% and 93.70% respectively for Top-1 and Top-5 [36]. Fu, Chen, and Li, in 2017, obtained an even better accuracy of 78.5% and 94.1% for Top-1 and Top-5 with improved deep 50-layer ResNet[18]. CNN model performed better in classification vs traditional methods on Food 101 dataset

Martinel et al. came to the conclusion that DLL models used existing models without specific features of the food being taken into account. They used dataset Food 101,UEC Food 256 and UEC food 100 and developed a new CNN structure WISeR specific for food recognition which combined deep residual blocks and a part convolution unit for extracting common vertical characteristics of food. Datasets achieved Top-1% and Top-5% rates of 90.27% and 98.71%, 83.15% and 95.45%, and 89.58% and 99.23% respectively[31]. In 2017, Mezgec and Seljak also developed a modified version of the AlexNet architecture, known as ‘NutriNet’ for food and drink classification. They explored both the training set and testing set and reached an accuracy rate of 86.72% and 94.47% respectively [44].

The most successful work in Food nutrients estimation was by Parisa Pouladzadeh aet al. which took into account color, size, shape and texture features in the data preparation for SVM, and then applied the RBF kernel, and lastly used SVM for classification[52]. They applied their method on categories of food , Three different types: single, non-mixed, and a combination or type of mixed foods, resulting in SVMs accuracy of around 92.21%, 85%, and 35%–65%, respectively[52].

Now, greater difficulties lie in food calorie estimation in comparison to food classification. The colour, texture, shape of the food are not the only factors that greatly influence the results but also, the ingredients and their quantities, cooking methods, cooking methods, and weights (or volumes) of each ingredient labelled with the respective calorie content all play a part in determining a fair result.

Chapter 3

Final Design of the System

3.1 Architectures chosen for this project

Inception v3 Architecture

Efficiency in the use of computation power with optimal cost is the core of any Inception network. In the Inceptionv3, this has been taken care of as well: [110] [66]

1. **Factorization of Convolutions**: In the Inception v3, the factorisation of convolutions to smaller sizes is key, it has helped to reduce parameters and connections resulting in less usage of computational power and greater resultant savings. In this process, it does not compromise on efficiency of the network. This means that bigger convolutions can be replaced with smaller convolutions, this still leads to faster training. The convolution size reduction may however, impact the performance to some extent [110].

2. **Asymmetric convolutions:** As explained earlier, making smaller convolutions is a key aspect of Inception v3. While making a smaller convolution, further downsizing convolution possibilities can be considered too. A 3×3 is not always the best option, it can be further improved to 2×2, but the introduction of two layer solutions like the combination of a 3×1 convolution followed by 1×3 proves to be cheaper and more efficient in some grid sizes. This essentially means that using asymmetric convolutions proves to be more efficient at times.
3. **Auxiliary classifier:** Auxiliary classifiers are added to the layer to make convergence of the network more stable. These are added in the end for deep networks and efficient convergence; introduction of Batch Normalisation improves this process even more and makes use of gradients as fast as possible.

4. **Grid size reduction:** Similar to any other convolutional network, Inceptionv3 also follows the optimisation of features with a pooling operation. In this process it reduces the grid size , achieves efficiency in the network and minimises cost. For example, 2 collections of feature maps are instead added as 640 feature maps in order to proceed to the next inception layer (Shown in Fig. 7 below).

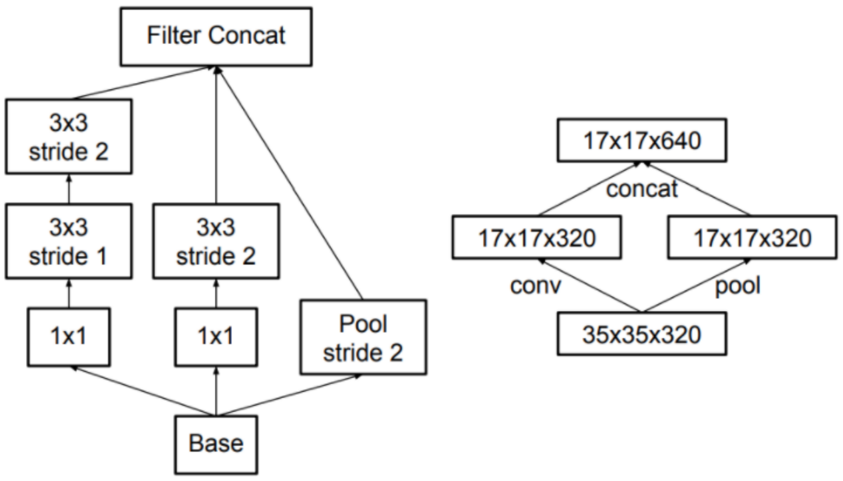
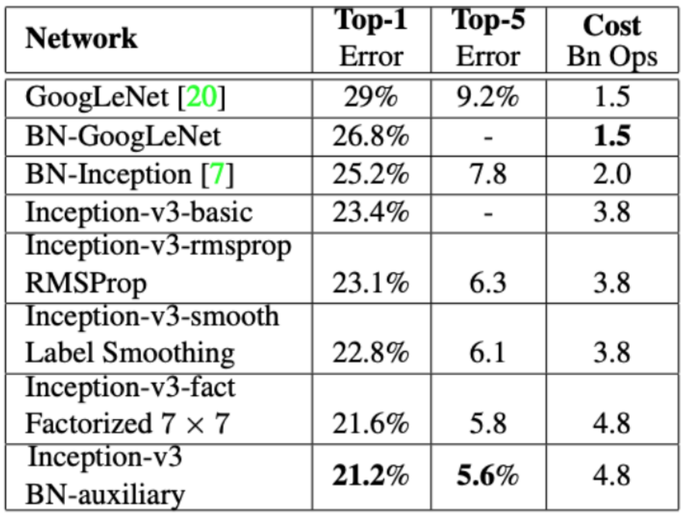


Fig. 7: Structure of Inceptionv3 architecture (Sourced from: [110])

Inception v3 Training and Results

The table below(in Fig. 8) depicts and compares the best results achieved when testing Inception models after training them on ImageNet. ImageNet dataset is the basis of Inception v3 which makes training efficient. Even though it can be trained on other datasets as well, with ImageNet as the base data, the results of the models can be checked with other models. This can be seen here in the picture. Along with factorisation into small convolutions, Auxiliary classifiers, Batch Normalisation, Label smoothing, RMS prop the best error rates can be seen in Inceptionv3 vs. other models [65].

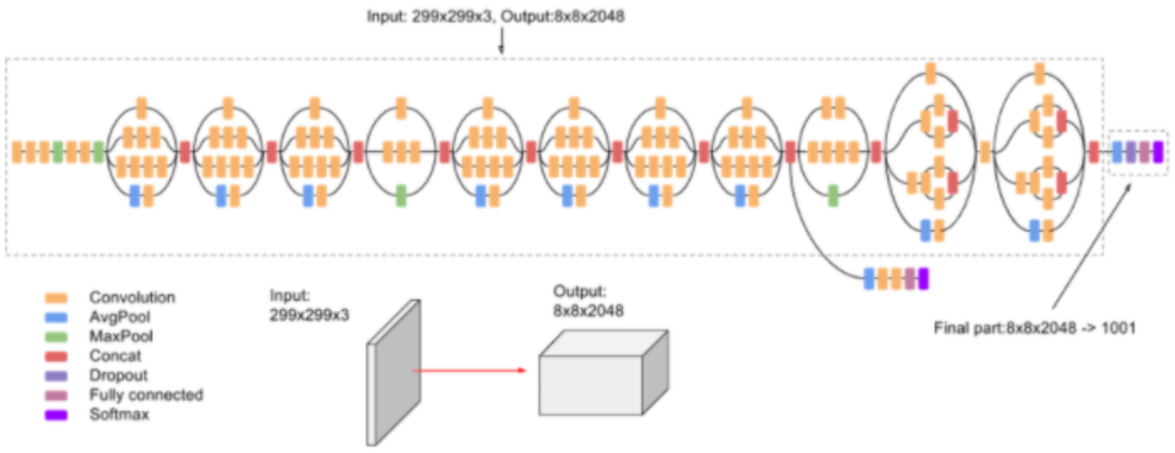


Network	Top-1 Error	Top-5 Error	Cost Bn Ops
GoogLeNet [20]	29%	9.2%	1.5
BN-GoogLeNet	26.8%	-	**1.5**
BN-Inception [7]	25.2%	7.8	2.0
Inception-v3-basic	23.4%	-	3.8
Inception-v3-rmsprop RMSProp	23.1%	6.3	3.8
Inception-v3-smooth Label Smoothing	22.8%	6.1	3.8
Inception-v3-fact Factorized 7 × 7	21.6%	5.8	4.8
Inception-v3 BN-auxiliary	**21.2%**	**5.6%**	4.8

Fig. 8: Different Inception models' performance (Sourced from: [110])

Inception-ResNet-v2 Architecture

As the name suggests it is a hybrid model comprising a basic platform of Inception without filter concatenation and includes residual connections. This results in enhanced accuracy in the ILSVRC image classification benchmark. This facilitates training on deep structure. It is an improved version of Inceptionv3 . Inclusion of residual connections simplifies Inception blocks and improves the performance and efficiency of the model; it can also train even deeper neural networks with simplified inception blocks [113] [114] [115].



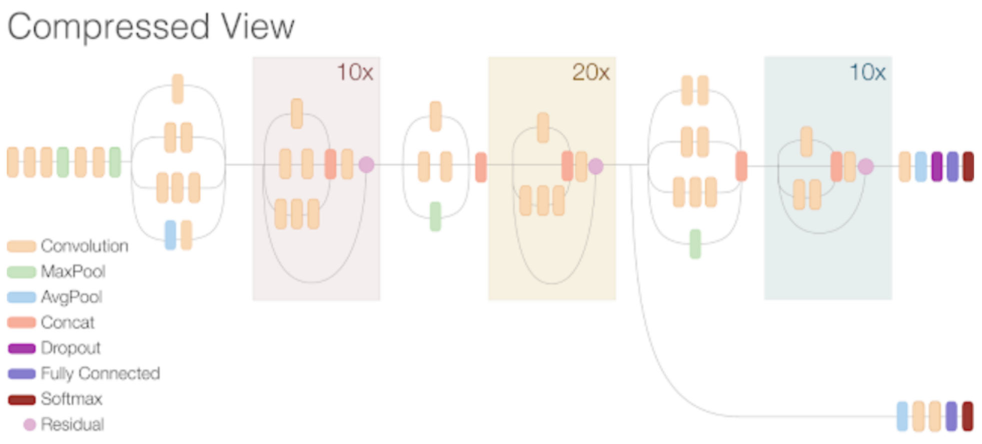


Fig. 9: Structural comparison of Inceptionv3 vs InceptionResNetv2 respectively (Sourced from: [116])

The Inception-ResNet-v2 figure is a compressed view of the repeating blocks in the architecture; the fully expanded network contains significantly more layers than the Inception V3(Models can be compared in Fig. 9). It also should be noted that the Inception blocks have been simplified in the InceptionResnetv2, it has fewer number of aligned towers compared to the Inception V3. The hybrid architecture is also more accurate than contemporary models, as shown in the table above , which reports the Top-1 and Top-5 validation accuracies on the ILSVRC 2012 image classification benchmark. It is important to note, however, that the new model only requires about twice the memory and computational capacity compared to Inception V3[67] [117].

In the table above on image benchmark the comparison of Res Net 2 with other models was done which shows the Top 1 and Top validation Accuracy level on the image; it shows Inception ResNetv2 efficiency is highest vs other models.

3.2 Datasets choice

The project aims to investigate existing papers on Object detection and classification as well as Food detection and classification, and existing food datasets (i.e. Food images in ImageNet[26] , food-11[69] , food-101[15].

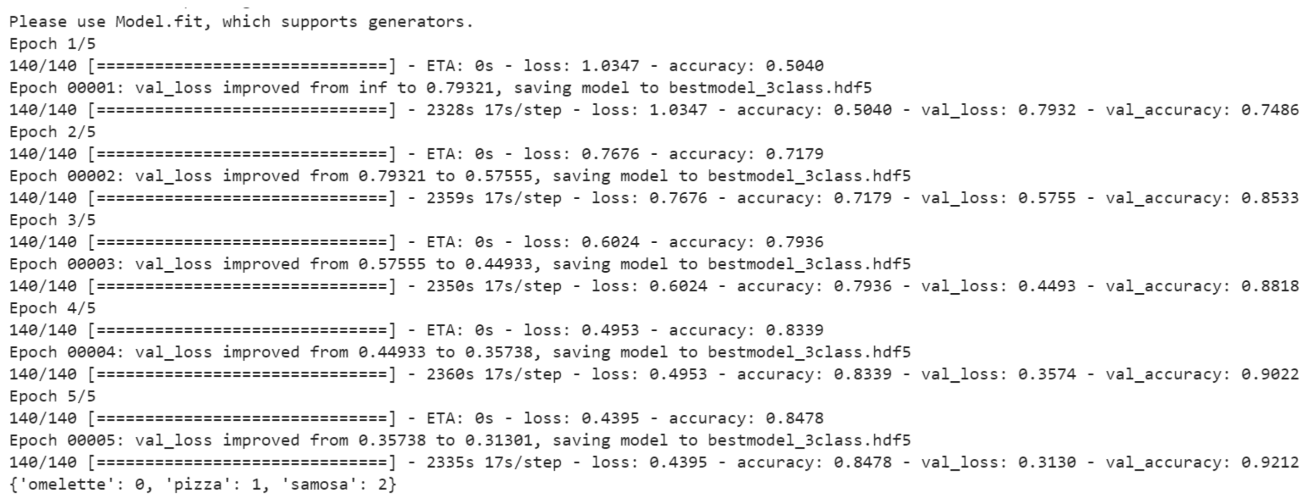
In the selection of the food database the key points considered were a large population of images so that machine can be trained in identification and classification with least error, diversified according to different taste and ethnic group, it should be predominately dealing with food items, with all these requirement dataset 101 was best suitable . as it has more than 100000 images including 750 training images and 250 manually reviewed test images, in more than 100 categories. Side length of images was restricted to 512 pixels[119].

The Food-11 dataset contains 16643 food images. These images are divided into 11 leading categories. More specifically, it has 3 folders: evaluation, training and validation with 9866, 3430 and 3347 images respectively. Each image is of the size 32*32. The number of images in each food class varies as well; for example there are 1500 images in the dessert and soup class, whereas there are only 280 in the rice class. A datagenerator is used for the images in each of the three folders to resize each image to 256*256 and make sure that the images are processed and fed into the deep learning model in batches(each of size 32). The class mode is categorical, meaning that the food class will be identified depending on the category it belongs to.

3.3 Technical Experimentation

The Neural network chosen for this project was an open source neural network and it was supported with the Tensorflow backend. Inceptionv3 has been chosen as the primary deep learning model for experimentation to date. The process followed is as follows: First of all, images from the Food-101 image dataset by ETH Zurich had been extracted from a tar.gz file. All the class titles of images inside the food-101/images folder were listed, following which the images were visualised - an image from each class was displayed and the dimensions were each checked to be 25*25.The data was split into separate train and test folders with 3:1 ratio respectively. Three classes were picked out of the 101 possible classes, namely samosa, omelette, and pizza and a dataset solely for these three classes was prepared accordingly.

There were several steps taken in training of the model, including data augmentation on all train, test and validation set of images; adjusting the shear range and zoom range to 0.2 as well as performing a horizontal flip. The weights from the imagenet dataset were frozen, but the top layer was excluded. Instead a new layer constructed from the GlobalAveragePooling2D function, followed by Dense and lastly by Dropout, was defined. The predictions were then defined to be outputted using the activation function softmax. The model was compiled and the model training function, when called prints the epoch (until 5), runtime, loss, accuracy, validation_loss and lastly validation_accuracy which after the last epoch reaches 92.12%. The results of training the model for different epochs can be seen below:



```
Please use Model.fit, which supports generators.
Epoch 1/5
140/140 [==============================] - ETA: 0s - loss: 1.0347 - accuracy: 0.5040
Epoch 00001: val_loss improved from inf to 0.79321, saving model to bestmodel_3class.hdf5
140/140 [==============================] - 2328s 17s/step - loss: 1.0347 - accuracy: 0.5040 - val_loss: 0.7932 - val_accuracy: 0.7486
Epoch 2/5
140/140 [==============================] - ETA: 0s - loss: 0.7676 - accuracy: 0.7179
Epoch 00002: val_loss improved from 0.79321 to 0.57555, saving model to bestmodel_3class.hdf5
140/140 [==============================] - 2359s 17s/step - loss: 0.7676 - accuracy: 0.7179 - val_loss: 0.5755 - val_accuracy: 0.8533
Epoch 3/5
140/140 [==============================] - ETA: 0s - loss: 0.6024 - accuracy: 0.7936
Epoch 00003: val_loss improved from 0.57555 to 0.44933, saving model to bestmodel_3class.hdf5
140/140 [==============================] - 2350s 17s/step - loss: 0.6024 - accuracy: 0.7936 - val_loss: 0.4493 - val_accuracy: 0.8818
Epoch 4/5
140/140 [==============================] - ETA: 0s - loss: 0.4953 - accuracy: 0.8339
Epoch 00004: val_loss improved from 0.44933 to 0.35738, saving model to bestmodel_3class.hdf5
140/140 [==============================] - 2360s 17s/step - loss: 0.4953 - accuracy: 0.8339 - val_loss: 0.3574 - val_accuracy: 0.9022
Epoch 5/5
140/140 [==============================] - ETA: 0s - loss: 0.4395 - accuracy: 0.8478
Epoch 00005: val_loss improved from 0.35738 to 0.31301, saving model to bestmodel_3class.hdf5
140/140 [==============================] - 2335s 17s/step - loss: 0.4395 - accuracy: 0.8478 - val_loss: 0.3130 - val_accuracy: 0.9212
{'omelette': 0, 'pizza': 1, 'samosa': 2}
```

Fig. 10: Greater accuracy with increasing number of epochs

3.4 The Implementation

Traditional approaches (OpenImaj, Java)

The following section explores the accuracy reached by deploying some of the mentioned techniques. In the beginning simple methods were applied. First initially started with nearest neighbour classification, tiny images, and gradually progressed using techniques like Bag of quantized local features, linear classifiers learned with Support Vector Machines.

<u>Model class</u>

The structure of the package has been set up such that it is logical and avoids code duplication. It contains the protected keywords trainingData of type VFSGroupDataset<FImage> and testData of type VFSListDataset<FImage> respectively, as well as a splitter of type GroupedRandomSplitter which is used to split the labelled training set into training and testing. The VFSGroupDataset is essentially a grouped dataset of VFSListDatasets. The benefit of using a GroupedRandomSplitter is that it allows the splits to be computed at any time. This makes it easy to generate new splits(for cross-validation for example). During this experimentation phase, both models(Test 1 and Test 2) were trained solely on the Food-11 image dataset.

<u>Test 1</u>

The K-nearest-neighbour classifier uses the 'tiny image' feature. Which is very simple image representations where image is cropped to a square about the centre. and then resized to a small, fixed resolution (16*16). The pixel values can be packed into a vector by concatenating each image row. Its performance improves when tiny images have zero mean and unit length. As a general rule, it was found that the greater the ratio allocated to the training set in comparison to the test set, the greater the reliability of the model and hence the higher the accuracy on the test set.

A discrete-valued function or hypothesis to assign categorical class labels to a particular data point is called classifier. The classifier is based on a multi-class k-nearest-neighbour classifier which brute-forces k-nearest-neighbour implementation for objects with a compatible DistanceComparator. It takes as arguments an instance of the Flattener class, FloatFVComparison, which compares the clusters by Euclidean distance, k clusters, as well as a splitter which splits the training set to training and testing with a 4:1 ratio. The instance comes from the inner class Flattener which, using the extractFeature method, returns the computed feature vector for each image, flattening each image into the vector of pixels it will use. It crops the image to a square about the center, shrinks the image to the chosen resolution, makes sure to keep the same aspect ratio and normalises it so that it has 0 mean and unit length, following which it flattens the pixel matrix into a vector.

In the min-max sealing, normalisation reduces a numerical distribution in the 0 to 1 interval. This numerical distribution change in the 0-1 interval brings the mean between 0 to 1. Instead of this in case of Zero centering values in the distribution are changed to make the mean as 0. Both of these techniques combined are known as standardization. Standardization achieves invariance to irrelevant differences by focusing on absolute and not relative values. The model was relieved from having to learn these differences. Moreover, it makes optimization of the model easier and faster by making it computationally less expensive.

After the model is run using classifier.trainMultiClass, the accuracy of the model is reported after performing cross-validation, and the results on the unlabelled test dataset are printed in a txt file in the format required. Prior to keeping a record of the final results, the hyperparameters were changed and observed to find the best values. The accuracy of the classifier was explored for different values of k, but no pattern was observed on the relationship between k and the accuracy rate. However, the maximum accuracy rate reached was 27.7% at k=11. The effect of altering the image resolution was further tested on the accuracy at this k value. More specifically, an accuracy of 21.8% was achieved when the resolution was set to 4 and it peaked at 34.1% when resolution was set to 32, but at a resolution greater than this, it declined again. A very high resolution suggests that there are too many pixels per inch to account for in the image but the image is of higher quality, whereas an image of a resolution too low means there are fewer pixels than required resulting in a lower quality and perhaps blurred image.

Test2

A set of linear classifiers (specifically the Liblinear Annotator class to automatically create 11 one-vs-all classifiers) are developed using a bag-of-visual-words feature based on fixed size densely-sampled pixel patches. The exploration started with 8x8 patches, sampled every 4 pixels in the x and y directions. A sample of these are clustered using K-Means to learn a vocabulary (starting with 500). Mean-centring and normalising of each patch was performed before clustering/quantisation. The pixels from the patches are then taken and flattened into a vector following which vector quantisation is used to map each patch to a visual word.

Each image is divided into patches which are in turn transformed into patch vectors. The type LocalFeatureList<FloatKeypoint> represents the patch vectors for all images. FloatKeypoint has a FloatPixelVector and the rectangle dimensions which suggest where the patch occurs in the image. RectangleSampler slides through the images and gives you bounds to take the patches from. The patches are randomised using Collections.shuffle(all the generated patches in an image) and then the 30 are picked using allPatches.sublist(30). The accuracy was measured both without shuffling the patches and after shuffling the patches. The resulting finding was that shuffling raises the accuracy by approximately 5%. The developed understanding is that if there are 100 pixels in an image and a sample of size 30 must be taken without shuffling, essentially just taking the top 3 rows of the image so it is not going to be representative of the features of the entire image.

Then, a K-Dimensional tree ensemble with 500 clusters passed into the constructor. There was a significant increase in accuracy from k = 300 to k = 500, and then a drop for k = 700. A HardAssigner is then created for the clusters and used to initialise the LiblinearAnnotator classifier. The classifier is then trained on the training set. All the patch vectors of the image are retrieved by the getPatchVectors method some of which are then randomly sampled using getPatchSamples method.extractROI will get the actual patch of the image (a sub image of the image) represented by the bounds of the rectangle. Each patch is mean-centered and normalised. A decrease in sep size results in a significant increase in accuracy, particularly it was found to increase from 56.7% to 65.3% with a decrease in step size from 8 to 2, and similarly there was an increase in accuracy from 66.1% to 71.0% with a decrease in patch size from 12 to 4. The WordsExtractor class implements FeatureExtractor; a new instance of the BagofVisualWords is initialized in the constructor and in the extractFeature method includes an aggregator - an instance of the BlockSpatialAggregator. The result of the aggregator is returned by the BlockSpatialAggregator.

The convenience class LocalFeatureListDataSource is used as multiple lists of features are fed into our k-means clustering algorithm. The FloatKeypoints in the list contain not only the vector of pixels of each patch but also their respective locations within the original image which are used by the clustering algorithm. The images are classified using densely sampled image patches and then grouped into a Bag of Visual Words and produce histograms of each image. As a result, the spatial aggregator is used to create those histograms.

For each image in the testing set, the model classifies the image and then writes the name of the image file followed by it's predicted class. The serializeResults method in the Results Serializer performs the necessary I/O, and saves the results to a file.

Deep Learning approaches

(Deep Learning libraries, Python)

Deep Neural Networks such as VGG, Inception andResNet have been trained on widely available datasets like Imagenet and the trained versions of these models are available for common use. We can use the learnt weights previously alongwith with the support of a few layers over the surface layer to fine-tune the model to our new data. This saves time, computation, and helps achieve convergence faster in comparison to models trained from the beginning.

There are two different approaches of transfer learning that we must choose between:[54]:

● Approach 1: This focus is put on training dense layers which are newly added with weights randomly initialised. The original CNN network weights from the initial layers which are already trained are not touched or fixed.

● Approach 2: Reset the CNN network with the predetermined weights. Train the whole CNN network once again while choosing a significantly lower learning rate; this is critical to ensure that the trained weights are not changed abruptly.

Transfer learning provides fast training progress, and prevents the need to train the model from scratch using randomly initialized weights. A small training dataset to achieve incredible results.

Inception-Resnet v2

The model is the result of the function with the basemodel(frozen model) as input and the headmodel(all layers that were added step by step until now) as the output. There are a total trained parameters are 54, 256, 192 out of a total 54, 336, 736 parameters . Binary cross entropy works for the loss function and for the optimizer Adam will be used. The key performance metric will be the accuracy level.

Steps taken to add classification head to the model:

1) Headmodel = Basemodel.output
2) Add GlobalAveragePooling2D layer
3) Headmodel = Add Flatten layer
4) Headmodel = Add (1st) Dense layer; 256 units, activation function:relu
5) Headmodel = Add Dropout layer with rate 0.3 to current headmodel
6) Headmodel = Add (2nd) Dense layer; activation function: relu has 128 units
7) Headmodel = Add Dropout layer with rate 0.3 to current headmodel
8) Headmodel = Add (3rd) Dense layer, 11 units with activation function: softmax

When compiling the model, loss is defined as categorical_crossentropy(as different categories are being identified) and optimized to be SGD(Stochastic Gradient Descent) with learning rate 0.01 and momentum 0.9. Define metrics being measured as accuracy. Early stopping is used to exit training if validation loss does not decrease even after certain epochs (defined by patience). The particular model saved in weights.hdf5 is the one with the lowest validation loss. This function is verbose. One of the automatic functions that is registered when training all deep learning models that perform at any training stage is the History callback. It keeps track of training metrics for each epoch, including the loss and accuracy for classification problems as well as the validation dataset, if either is set. The history object is called back from the fit() function, which is specifically used to train the model. Metrics are stored in the dictionary of the history of the returned object. Plots are generated from the composed history data.

When compiling the model, loss is defined as categorical_crossentropy(as different categories are being identified) and optimized to be SGD(Stochastic Gradient Descent. Here the momentum is 0.9 and the learning rate of 0.01. Define metrics being measured as accuracy. In the case of loss from validation after a certain epoch, do not reduce, then for exit training early stopping is done (defined by patience). Only models having minimum validation loss are considered fit for saving in the weights.hdf5 file. This function is verbose.

History callbacks are the most common callbacks used in most of the deep learning models. In this callback , training models return a history object from calls to the fit ()function being used . Every epoch's training matrix is being recorded. It also records validation dataset accuracy including loss, same for classification issues[121].

History can be generated by fitting the train generator, passing in steps = test_generator.n // 32, the number of epochs and validation_data. Load weights from weights_fine.hdf5. To evaluate the model, call evaluate_generator with arguments test_generator, steps = test_generator.n // 32, and verbose = 1. Labels given to the different classes. Empty arrays of prediction, original, and image, and count = 0. Each img item in the evaluation folder is resized to (256, 256) and appended to the image array. The image is then normalized and reshaped to the required dimensions (-1, 256, 256, 3). The model used model.predict to predict, and then get the index corresponding to the highest value in the prediction. The predicted and original classes are then appended to the prediction list. The values in the prediction and original array are checked to measure the accuracy of the prediction. The classification report is printed using the original and prediction array. Finally, a confusion matrix illustrates the predicted values as x-label and original values as y-label.

The example below records history, returns from training the model and puts these charts(shown in Figure 11):

A plot of accuracy and loss on the training and validation datasets over training epochs, respectively.

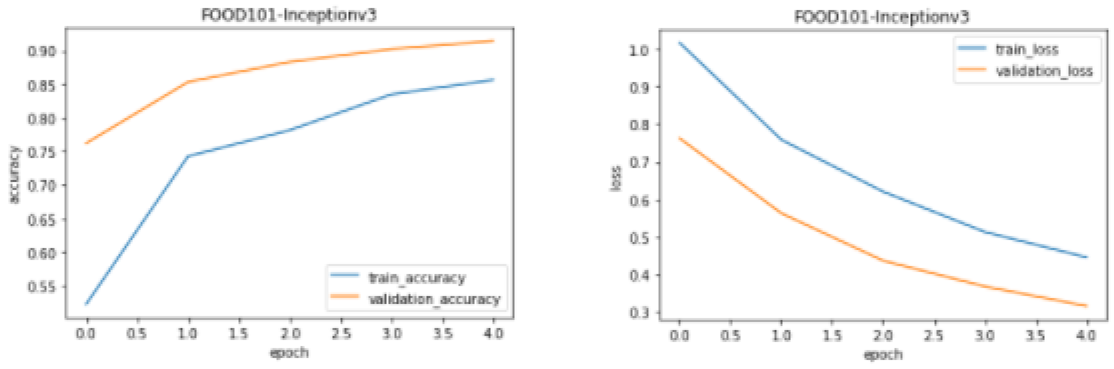


Fig. 11: Train, Validation Loss and Accuracy charts[122]

<u>Inceptionv3; Food-101</u>

The tensorflow/examples repo (with images to evaluate a trained model) has been cloned. The data has been downloaded and extracted, and randomly images have been selected to view from respective classes. The image data is used for training and testing with the help of train/test txt (copy images from food-101 or food-11 using the file test.txt). Only a subset of data with few classes (11) is used .To experiment and try different architectures, train_min and test_min limit the dataset to 11 classes. Experimenting with all 101 classes would take too much time and computation.The image width and height to be 299, 299 and batch size have been defined, and class mode has been set to categorical, following which the train dataset is augmented. The data augmentation performed is identical to the Inception-Revsnetv2 model. The final fine-tuned Inceptionv3 is the output resulting from feeding the inceptionv3 model(with weights from imganet and excluding the top layer) into the function. A GlobalAveragePooling2D layer was added in the model which was fixed. This was further supplemented with the Activation function/ReLU, a dense layer of 128 units and a dropout layer having 0.2 rate. The predictions are recorded by applying a final sense layer and passing in the number of classes, the kernel regularizer, and activation as softmax. Compilation of this model was supported by the SGD optimizer having a momentum 0.9 learning rate of 0.001.Define loss as categorical_crossentropy, and metrics as accuracy. The best results are saved at the checkpoint. Following this, accuracy and loss graphs against epochs have been plotted. Each graph will show the result of the train and validation respectively.

The plots show that the accuracy of the model increased with epochs whereas the loss has decreased. Validation accuracy has, however, been on the higher side than training accuracy for many epochs. This can be due to two reasons; the pretrained model trained on ImageNet which contains data from a variety of classes or using dropout can lead to a higher validation accuracy.

The conclusion we can draw is that with epochs accuracy level increases and on the other side helps in reducing or minimising the loss. We can also observe that compared to training accuracy in some epochs the accuracy of validation is more. It is possible this change in outcome may be due to the dropout that was used which in turn increased the accuracy and another model was trained on imagenet having a large and diverse variety of classes.

The model proved to be ideal and saved to make predictions; setting compile =False and clearing with this model seems to be loaded more rapidly; otherwise in the absence of these parameters the loading of the model would take more time. In order to predict the images through a model, the images are loaded one by one, and changed to a target size. Following this, the image is converted into an array, the dimensions are expanded (made 3-dimensional). Upon running the model, each epoch took around 7 minutes to run. While the loss was still decreasing, the model could be trained for some more epochs. Similar to the Inception-Resnetv2 model,

Time taken in training of models is much higher in the case of 101 classes; i.e. more than an hour for each epoch. In comparison, the model performed quite well with eleven classes.

Hyper-parameter and model tuning

The optimal train-test split in each of the models depends both on the dataset the model is being trained on and the model itself. Usually deep learning models require the most percentage of the dataset to be used for training as a small percentage of the training set will result in an unreliable, underfitting model. Conversely, a very small test set can lead to the overfitting of the data.The train and test splist chosen for this project are 7:3, 8:2 and 9:1 respectively.

Learning algorithms cover whole training sets , though it is an epoch which fixes the number of times the algorithm will cover this. Every epoch can have even more than one batch and every image in the training data set has the option to change the internal parameters of the model. An epoch can be visualised like a for-loop for some epoch, while loops are proceeding over the training data set. The batches of samples are passed through by loops with these for-loops (referred herein as batch size) [7].

The number of epochs is often very large, perhaps even hundreds, permitting the learning algorithm to run until the error from the model has diminished.The learning rate of the model can be depicted using learning plots/curves that show epochs along the horizontal x-axis as time and the model error on the vertical y-axis. These assist in determining whether the model has over-learnt, under-learned, or fits the training dataset in an optimal manner. The number of epochs chosen for this project are 1, 5 and 10. The number of epochs is often very large, perhaps even hundreds. These Epochs facilitate the running of learning algorithms to make models with least error. The process of learning of the model can be shown in the graph with the help of Learning plots/Curves where we can put on the X axis Epochs showing them as time correspondingly in the Y axis we can put model errors.These assist in determining whether the model has over-learnt, under-learnt, or fits the training dataset in an optimal manner. The number of epochs chosen for this project are 1, 5 and 10. [7].

The size of a batch is the image sample count calculated prior to the model Size of the batch is very important , it should be either matching the numbers of samples of the training dataset or less . It should be atleast exceeding or equal to one in the training dataset.

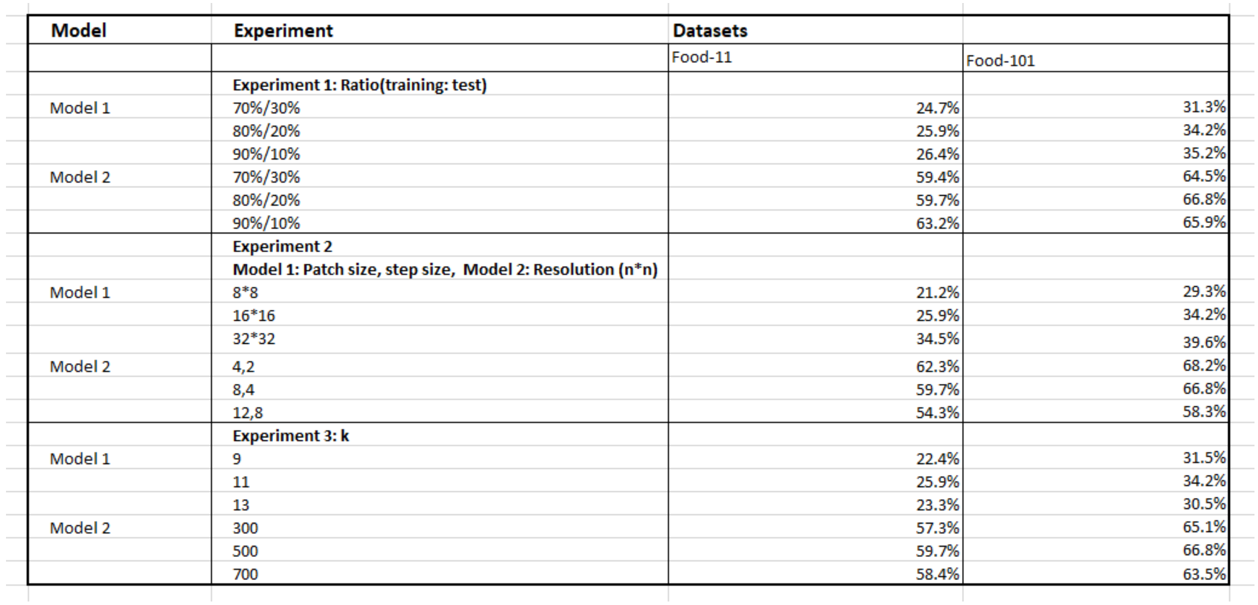
Consider a scenario where we have a dataset with 5000 samples and a batch size of 50 and 10 epochs is chosen. In this case it means we can make one epoch out of 100 batches or updates, and every batch has 50 samples. Moreover it means every epoch which deals with 100 batches or makes 100 changes in the model. As explained earlier, 10 epoches means the model will pass the data sets 10 times. The whole training process will be 1000 batches.in this process the weights need to be modified once every batch of 50 is done [7].

The method which can be used for calculation of the Error gradient is SGD or Stochastic Gradient Descent. This method calculates the error gradient for the present model with samples of training datasets. In this process this method uses back- propagation to change the weight . This quantum of change in weights at the time of training can be termed as learning rates . the range of learning rate scan be 0.0 to 1.0 here 0.01 shows variation in the weight by (0.1 * weight error estimate))[8].

In a specified number of training epochs , with optimal learning rates, the model utilises the resources like number of layers, number of nodes for each layer etc to learn in the best possible manner and makes an optimal approximation of function. Usually, a higher learning rate permits the model to learn more quickly, with the cost of reaching a poor set of final weights. A lower learning rate may however, allow the model to learn better or perhaps a global choice of weights too. This may also take notably longer to train.

The performance of the model to a great extent depends on learning rates , larger learning rate will have large changes in weight which may be divergent also, may impact the efficiency of model(like loss on the dataset of training) will fluctuate on the training epoch. But at the same time a small learning rate has its own issues. It may face a problem in convergence or the solution itself may not be so ideal or sub-optimal. In some worse situations the proportion of weight needs to be calibrated very carefully as large updates can create numerical overflow resulting in the explosion of the weights. The rates chosen for this project are 0.1, 0.01 and 0.001 [8].

3.5 Results



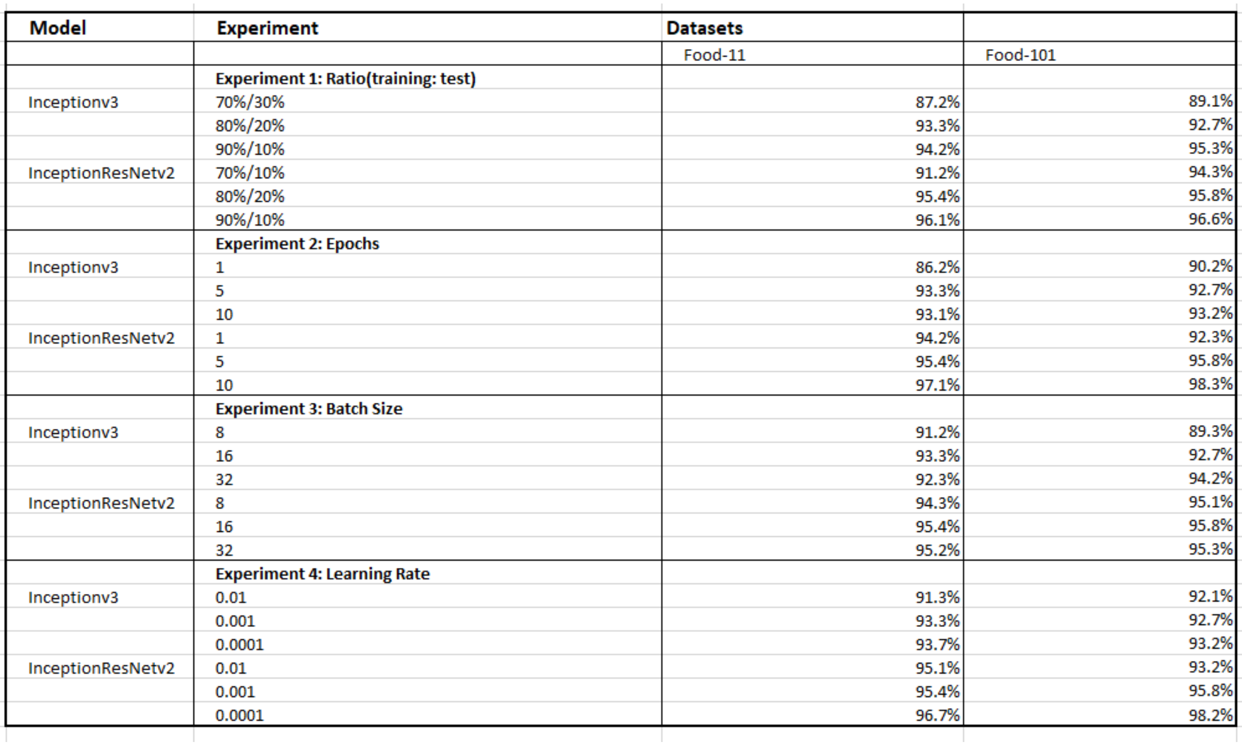
Model	Experiment	Datasets	
		Food-11	Food-101
	Experiment 1: Ratio(training: test)		
Model 1	70%/30%	24.7%	31.3%
	80%/20%	25.9%	34.2%
	90%/10%	26.4%	35.2%
Model 2	70%/30%	59.4%	64.5%
	80%/20%	59.7%	66.8%
	90%/10%	63.2%	65.9%
	Experiment 2		
	Model 1: Patch size, step size, Model 2: Resolution (n*n)		
Model 1	8*8	21.2%	29.3%
	16*16	25.9%	34.2%
	32*32	34.5%	39.6%
Model 2	4,2	62.3%	68.2%
	8,4	59.7%	66.8%
	12,8	54.3%	58.3%
	Experiment 3: k		
Model 1	9	22.4%	31.5%
	11	25.9%	34.2%
	13	23.3%	30.5%
Model 2	300	57.3%	65.1%
	500	59.7%	66.8%
	700	58.4%	63.5%

Fig 12: Results - Average of 3 Runs

(Traditional Approaches)

Model 1: K-NN classifier using tiny-image feature

Model 2: Set of Linear Classifiers



Model	Experiment	Datasets	
		Food-11	Food-101
	Experiment 1: Ratio(training: test)		
Inceptionv3	70%/30%	87.2%	89.1%
	80%/20%	93.3%	92.7%
	90%/10%	94.2%	95.3%
InceptionResNetv2	70%/10%	91.2%	94.3%
	80%/20%	95.4%	95.8%
	90%/10%	96.1%	96.6%
	Experiment 2: Epochs		
Inceptionv3	1	86.2%	90.2%
	5	93.3%	92.7%
	10	93.1%	93.2%
InceptionResNetv2	1	94.2%	92.3%
	5	95.4%	95.8%
	10	97.1%	98.3%
	Experiment 3: Batch Size		
Inceptionv3	8	91.2%	89.3%
	16	93.3%	92.7%
	32	92.3%	94.2%
InceptionResNetv2	8	94.3%	95.1%
	16	95.4%	95.8%
	32	95.2%	95.3%
	Experiment 4: Learning Rate		
Inceptionv3	0.01	91.3%	92.1%
	0.001	93.3%	92.7%
	0.0001	93.7%	93.2%
InceptionResNetv2	0.01	95.1%	93.2%
	0.001	95.4%	95.8%
	0.0001	96.7%	98.2%

Fig 13: Results - Average of 3 Runs

(Deep learning approaches)

Chapter 4

Conclusions

● Performance of all models is better on the Food-101 dataset compared to Food-11 (at most times). This is perhaps due to the fact that there are a fixed number of images (1000) in each class of the prior dataset, whereas the number of images can vary from 200 to 1500 in the latter dataset.

● Deep Learning models are much more accurate (lowest accuracy 86.2 %) compared to 21.2% in traditional approaches.

● Performance of the Inception-Resnetv2 model surpasses the Inception-v3 model in most cases.

● Performance of the Set of Linear Classifiers model surpasses the K-NN classifier using the tiny-image feature model in all cases.

● Traditional Approach Model 1:

★ Different resolution: Decrease in resolution -> decrease in accuracy.

★ Different values of k: Increase in k -> increase in accuracy from k = 9 to 11 but decline from k=11 to 13.

● Traditional Approach Model 2:

★ Different step size: Decrease in step size -> decrease in accuracy.

★ Different patch size: Decrease in patch size -> increase in accuracy.

★ Different values of k: Increase in k -> increase in accuracy from k = 300 to 500 but decline from k=500.

- Deep Learning Approaches

    - ★ Different number of epochs: Increase in number of epochs -> increase in accuracy.

    - ★ Different batch size: Increase in batch size -> increase in accuracy.

    - ★ Different learning rate: Decrease in learning rate -> increase in accuracy.

- In comparison to existing literature, the results from this project are very much similar. The researcher who used the bag-of-SIFT approach to train his own dataset of 101 classes, reached an accuracy of 24%. This approach has resulted in an average accuracy of 29.4% across all the trials conducted in the two datasets for classification of 11 classes in this project. Moreover, Mustafa and Dhar reached an accuracy of 68.7% on their KNN model, making it very similar to the average of 63.6% reached by training our KNN model on the Food-11 and Food-101 datasets. The **highest accuracy** reached in this project was from the Inception-Resnetv2 model, of **98.3%** on the Food-101 dataset which **surpasses** the best Top-1 score of 94.5% from Mezgec and Seljak. The hypothesis is that this is mostly due to the nature of the model(Inception-Resnet v2 vs Alexnet) which has been improved over the years, but another project would need to be conducted to explore the specific reasons.

Chapter 5

Further Improvements

- Until now the datasets used have been constricted to 11 classes.Training the models on the entire dataset would be computationally very expensive but may produce better-trained models.
- Try different hyper parameters, modify their values in a larger specified range with smaller increments and explore how it impacts model performance.
- Set a threshold that the precision accuracy must cross when identifying the food. If the accuracy fails to cross this minimum threshold for its top prediction, the prediction can be considered equivalent to receiving no prediction at all.
- There is currently no implementation to handle a scenario where the image does not belong to any category out of all possible ones. Add a new class and feed in a new set of images that is significantly different to the original dataset content. This way the model also learns how to classify totally unseen/unrelated data as belonging to the new class.
- Both the Food-101 and Food-11 datasets are solely suitable for classification purposes. Usually, an object is placed near the food to find the relative size of the food in the image and to calculate its distance from the camera. The amount of food, its ingredient composition, and literature nutrient value content of the food in the image can be very different and difficult to calculate. The optimal way to address all three factors would be to create a new dataset of images, with all these factors being controlled variables.

Chapter 6

Bibliography

[1] A. Singla, L. Yuan and T. Ebrahimi, "Food/Non-food Image Classification and Food Categorization using Pre-Trained GoogLeNet Model", Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management - MADiMa '16, 2016. Available: 10.1145/2986035.2986039. [Accessed 8 December 2020].

[2] Arxiv.org. 2021. [online] Available at: <https://arxiv.org/pdf/1801.07239.pdf> [Accessed 14 September 2021].

[3] Baxter, J.. "Food Recognition using Ingredient-Level Features." (2012). Massachusetts Institute of Technology. https://www.semanticscholar.org/paper/Food-Recognition-using-Ingredient-Level-Features-Baxter/6cc1e3fedca5029b47dced0bb6beb26e33f6c819.[Accessed 6 December 2020].

[4] Bedford, Simon. "Simon Bedford." Simon Bedford Atom, simonb83.github.io/machine-learning-foodclassification.html. [Accessed 6 December 2020].

[5] Brownlee, J., 2021. A Gentle Introduction to Dropout for Regularizing Deep Neural Networks. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/> [Accessed 14 September 2021].

[6] Brownlee, J., 2021. A Gentle Introduction to Transfer Learning for Deep Learning. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/transfer-learning-for-deep-learning/> [Accessed 14 September 2021].

[7] Brownlee, J., 2021. Difference Between a Batch and an Epoch in a Neural Network. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/> [Accessed 14 September 2021].

[8] Brownlee, J., 2021. How to Configure the Learning Rate When Training Deep Learning Neural Networks. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/>

[Accessed 14 September 2021].

[9] Ciocca, G., Napoletano, P., & Schettini, R. (2017). Food recognition: A new dataset, experiments, and results. IEEE Journal of Biomedical and Health Informatics, 21(3), 588–598. https://doi.org/10.1109/jbhi.2016.2636441.[Accessed 6 December 2020].

[10] Clarifai, T., 2021. What Food Is This? Food Recognition Technology Can Tell You!. [online] Clarifai.com. Available at: <https://www.clarifai.com/blog/what-food-is-this-food-recognition-technology-can-tell-you> [Accessed 14 September 2021].

[11] "Convolutional Neural Network (CNN) | TensorFlow Core", TensorFlow, 2020. [Online]. Available: https://www.tensorflow.org/tutorials/images/cnn. [Accessed: 07- Dec- 2020].

[12]C. Szegedy, S. Ioffe, V. Vanhoucke and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning", arXiv.org, 2016. [Online]. Available: https://arxiv.org/abs/1602.07261v2. [Accessed: 07- Dec- 2020].

[13]C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision", arXiv.org, 2015. [Online]. Available: https://arxiv.org/abs/1512.00567v3. [Accessed: 07- Dec- 2020].

[14]Ege, T., & Yanai, K. (2018). Image-based food calorie estimation using recipe information. IEICE Transactions on Information and Systems, E, 101D(5), 1333–1341. https://doi.org/10.1587/transinf.2017MVP0027. [Accessed 6 December 2020].

[15]"Food101 | TensorFlow Datasets", TensorFlow, 2020. [Online]. Available: https://www.tensorflow.org/datasets/catalog/food101. [Accessed: 08- Dec- 2020].

[16]"Food Calorie Measurement and Classification of Food Images", International Journal of Pharmaceutical Research, vol. 12, no. 04, 2020. Available: 10.31838/ijpr/2020.12.04.262. [Accessed 7 December 2020].

[17]F. Ragusa, V. Tomaselli, A. Furnari, S. Battiato and G. Farinella, "Food vs Non-Food Classification", MADiMa '16: Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management, pp. 77-81, 2016. Available: https://dl.acm.org/doi/10.1145/2986035.2986041. [Accessed 8 December 2020].

[18]Fu, Z. H., Chen, D., & Li, H. Y. (2017). ChinFood1000: A large benchmark dataset for Chinese food recognition. In D. S. Huang, V. Bevilacqua, P. Premaratne, & P. Gupta (Eds.),

Intelligent Computing Theories and Application, ICIC 2017, Pt I (Vol. 10361, pp. 273–281). https://doi.org/10.1007/978-3-319-63309-1_25. [Accessed 5 December 2020].

[19] G. Ciocca, P. Napoletano and R. Schettini, "Food Recognition: A New Dataset, Experiments, and Results", IEEE Journal of Biomedical and Health Informatics, vol. 21, no. 3, pp. 588-598, 2017. Available: 10.1109/jbhi.2016.2636441. [Accessed 5 December 2020].

[20] G. Yiğit and B. Özyildirim, "Comparison of convolutional neural network models for food image classification", Journal of Information and Telecommunication, vol. 2, no. 3, pp. 347-357, 2018. Available: https://www.tandfonline.com/doi/full/10.1080/24751839.2018.1446236. [Accessed 8 December 2020].

[21]Heravi, E. J., Aghdam, H. H., & Puig, D. (2015). A deep convolutional neural network for recognizing foods. In Proceedings of 8th International Conference on 22 Machine Vision (Vol. 9875). https://doi.org/10.1117/12.2228875. [Accessed 8 December 2020].

[22]Heravi, E. J., Aghdam, H. H., & Puig, D. (2018). An optimized convolutional neural network with bottleneck and spatial pyramid pooling layers for classification of foods. Pattern Recognition Letters, 105, 50–58. https://doi.org/10.1016/j.patrec.2017.12.007. [Accessed 9 December 2020].

[23]Heravi, E. J., Aghdam, H. H., & Puig, D. (2017). Classification of foods by transferring knowledge from the ImageNet dataset. In Proceedings of 9th International Conference on Machine Vision (Vol. 10341). https://doi.org/10.1117/12.2268737. [Accessed 5 December 2020].

[24]H. Kagaya, K. Aizawa and M. Ogawa, "Food Detection and Recognition Using Convolutional Neural Network", 2020. Available: https://www.researchgate.net/publication/266357771_Food_Detection_and_Recognition_Using_Convolutional_Neural_Network. [Accessed 7 December 2020].

[25]I. Fadelli, "FoodTracker: An AI-powered food detection mobile application", Techxplore.com, 2020. [Online]. Available: https://techxplore.com/news/2019-09-foodtracker-ai-powered-food-mobile-application.html. [Accessed: 08- Dec- 2020].

[26]"ImageNet", Image-net.org, 2020. [Online]. Available: http://image-net.org/. [Accessed: 08- Dec- 2020].

[27]J. Hare, "COMP6248 Differentiable Programming (and Deep Learning)", Comp6248.ecs.soton.ac.uk, 2020. [Online]. Available: http://comp6248.ecs.soton.ac.uk/labs/lab6/. [Accessed: 07- Dec- 2020].

[28]J. Sun, K. Radecka and Z. Zilic, "Exploring Better Food Detection via Transfer Learning", 2019 16th International Conference on Machine Vision Applications (MVA), 2019. Available: 10.23919/mva.2019.8757886 [Accessed 7 December 2020].

[29]Kagaya, Hokuto & Aizawa, Kiyoharu & Ogawa, Makoto. (2014). Food Detection and Recognition Using a Convolutional Neural Network. 10.13140/2.1.3082.1120. [Accessed 6 December 2020].

[30]Kiourt, C., Pavlidis, G. and Markantonatou, S., 2021. Deep learning approaches in food recognition. [online] arXiv.org. Available at: <https://arxiv.org/abs/2004.03357> [Accessed 14 September 2021].

[31]Kiourt, C., Pavlidis, G. and Markantonatou, S., (2020), Deep learning approaches in food recognition, MACHINE LEARNING PARADIGMS - Advances in Theory and Applications of Deep Learning, Springer.

[33]Konaje, Nayan Kumar. “Food recognition and calorie extraction using Bag-of-SURF and Spatial Pyramid Matching methods.” [Accessed: 09- Dec- 2020].

[34]K. Yanai and Y. Kawano, "Food image recognition using deep convolutional network with pre-training and fine-tuning," 2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), Turin, 2015, pp. 1-6, doi: 10.1109/ICMEW.2015.7169816. [Accessed 8 December 2020].

[35]Liang-yc. "ECUSTFD_Bifrost Data Search", Datasets.bifrost.ai, 2020. [Online]. Available: https://datasets.bifrost.ai/info/230. [Accessed: 08- Dec- 2020].

[36]Liu C., Cao Y., Luo Y., Chen G., Vokkarane V., Ma Y. (2016) DeepFood: Deep Learning-Based Food Image Recognition for Computer-Aided Dietary Assessment. In: Chang C., Chiari L., Cao Y., Jin H., Mokhtari M., Aloulou H. (eds) Inclusive Smart Cities and Digital Health. ICOST 2016. Lecture Notes in Computer Science, vol 9677. Springer, Cham. https://doi.org/10.1007/978-3-319-39601-9_4. [Accessed 8 December 2020].

[37] Luo Juan, O., 2021. A Comparison of SIFT, PCA-SIFT and SURF. [online] Citeseerx.ist.psu.edu. Available at: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.301.7041> [Accessed 14 September 2021].

[38]L. Zhou, C. Zhang, F. Liu, Z. Qiu and Y. He, "Application of Deep Learning in Food: A Review", Comprehensive Reviews in Food Science and Food Safety, vol. 18, no. 6, pp. 1793-1811, 2019. Available: 10.1111/1541-4337.12492 [Accessed 7 December 2020].

[39]Martinel, N., Foresti, T. L., & Micheloni, C. (2018). Wide-slice residual networks for food recognition. In 2018 IEEE Winter Conference on Applications of Computer Vision (pp. 567–576). https://doi.org/10.1109/WACV.2018.00068. [Accessed 8 December 2020]. [Accessed 5 December 2020].

[40]M. A. Subhi and S. Md. Ali, "A Deep Convolutional Neural Network for Food Detection and Recognition," 2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES), Sarawak, Malaysia, 2018, pp. 284-287, doi: 10.1109/IECBES.2018.8626720. [Accessed 8 December 2020].

[41] Medium. 2021. Activation Functions in Neural Networks. [online] Available at: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6> [Accessed 14 September 2021].

[42] Medium. 2021. Architecture comparison of AlexNet, VGGNet, ResNet, Inception, DenseNet. [online] Available at: <https://towardsdatascience.com/architecture-comparison-of-alexnet-vggnet-resnet-inception-densenet-beb8b116866d> [Accessed 14 September 2021].

[43] Medium. 2021. Dense layers explained in a simple way. [online] Available at: <https://medium.com/datathings/dense-layers-explained-in-a-simple-way-62fe1db0ed75> [Accessed 14 September 2021].

[44]Mezgec, S., & Seljak, B. K. (2017). NutriNet: A deep learning food and drink image recognition system for dietary assessment. Nutrients, 9(7). https://doi.org/10.3390/nu9070657. [Accessed 2 December 2020].

[45]M. Subhi, S. Ali and M. Abdulameer, "Deep Convolutional Networks for Food Detection and Classification", Journal of Computational and Theoretical Nanoscience, vol. 16, no. 5, pp. 2433-2438, 2019. Available: 10.1166/jctn.2019.7913.[Accessed 2 December 2020].

[46]Myers, A., Johnston, N., Rathod, V., Korattikara, A., Gorban, A., Silberman, N., . . .Murphy, K. (2015). Im2Calories: Towards an automated mobile vision food diary. In 2015 IEEE International Conference on Computer Vision (pp. 1233–1241). https://doi.org/10.1109/ICCV.2015.146. [Accessed 3 December 2020].

[47]N. Mahony et al., "Deep Learning vs. Traditional Computer Vision", IMaR Technology Gateway, Institute of Technology Tralee, Tralee, Ireland, 2019. Available: https://www.researchgate.net/publication/331586553_Deep_Learning_vs_Traditional_Computer_Vision. [Accessed 8 December 2020].

[48]"Obesity - Causes", nhs.uk, 2020. [Online]. Available: https://www.nhs.uk/conditions/obesity/causes/. [Accessed: 08- Dec- 2020].

[49]Pandey, P., Deepthi, A., Mandal, B., & Puhan, N. B. (2017). FoodNet: Recognizing foods using ensemble of deep networks. IEEE Signal Processing Letters, 24(12), 1758–1762. https://doi.org/10.1109/lsp.2017.2758862. [Accessed: 03- Dec- 2020].

[50]Parisa Pouladzadeh, Abdulsalam Yassine, Shervin Shirmohammadi, August 1, 2020, "FooDD: Food Detection Dataset for Calorie Measurement Using Food Images", IEEE Dataport, doi: https://dx.doi.org/10.21227/yvk7-qk38. [Accessed: 07- Dec- 2020].Brownlee, JBrownlee, J

[51]P. McAllister, H. Zheng, R. Bond and A. Moorhead, "Combining deep residual neural network features with supervised machine learning algorithms to classify diverse food image datasets", Elsevier Ltd., 2018. Available: https://pubmed.ncbi.nlm.nih.gov/29549733/. [Accessed 8 December 2020].

[52]Pouladzadeh, P., S. Shirmohammadi and R. Almaghrabi. “Measuring Calorie and Nutrition From Food Image.” IEEE Transactions on Instrumentation and Measurement 63 (2014): 1947-1956. [Accessed: 03- Dec- 2020].

[53]R. Mustafa and P. Dhar, "A Method to Recognize Food using Gist and SBrownlee, JURF Features," 2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Kitakyushu, Japan, 2018, pp. 127-130, doi: 10.1109/ICIEV.2018.8641072. [Accessed: 07- Dec- 2020].

[54]"Transfer learning and fine-tuning | TensorFlow Core", TensorFlow, 2020. [Online]. Available: https://www.tensorflow.org/tutorials/images/transfer_learning. [Accessed: 07- Dec- 2020].

[55] Tyagi, Deepanshu. “Introduction to Sift(Scale Invariant Feature Transform).” Medium, Data Breach, 7 Apr. 2020, medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40.

[56]"What Is Food Culture And How Does It Impact Your Health?", The Well Essentials, 2020. [Online]. Available: https://www.thewellessentials.com/blog/what-is-food-culture-and-what-does-it-have-todo-with-our-health. [Accessed: 09- Dec- 2020].

[57]Yoshiyuki Kawano and Keiji Yanai, Automatic Expansion of a Food Image Dataset Leveraging Existing Categories with Domain Adaptation, Proc. of ECCV Workshop on Transferring and Adapting Source Knowledge in Computer Vision (TASK-CV), 2014.

[58]Z. Shen, A. Shehzad, S. Chen, H. Sun and J. Hiu, "Machine Learning Based Approach on Food Recognition and Nutrient Estimation", vol. 174, pp. 448-453, 2020. Available: https://www.sciencedirect.com/science/article/pii/S1877050920316331. [Accessed 8 December 2020].

[59]"Obesity and overweight", Who.int, 2020. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight. [Accessed: 09- Dec- 2020].

[60]"OpenIMAJ: Open Intelligent Multimedia Analysis", Openimaj.org, 2020. [Online]. Available: http://openimaj.org/tutorial/. [Accessed: 09- Dec- 2020].

[61]S. Saha, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way", Medium, 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed: 09- Dec- 2020].

[62]J. Brownlee, "A Gentle Introduction to the ImageNet Challenge (ILSVRC)", Machine Learning Mastery, 2020. [Online]. Available: https://machinelearningmastery.com/introduction-to-the-imagenet-large-scale-visual-recognition-challenge-ilsvrc/. [Accessed: 09- Dec- 2020].

[63]V. Mishra, "CNN Architecture: How ResNet works and why?", Medium, 2020. [Online]. Available:https://medium.com/datadriveninvestor/cnn-architecture-how-resnet-works-and-why-1c197b8eba34. [Accessed: 09- Dec- 2020].

[64] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[65]"A Simple Guide to the Versions of the Inception Network", Medium, 2020. [Online]. Available:https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202. [Accessed: 09- Dec- 2020].

[66]"A Guide to ResNet, Inception v3, and SqueezeNet | Paperspace Blog", Paperspace Blog, 2020. [Online]. Available:

https://blog.paperspace.com/popular-deep-learning-architectures-resnet-inceptionv3-squeezenet/. [Accessed: 09- Dec- 2020].

[67]"Improving Inception and Image Classification in TensorFlow", Medium, 2020. [Online]. Available: https://medium.com/@penolove15/improving-inception-and-image-classification-in-tensorflow-1e3c2ada4572. [Accessed: 09- Dec- 2020].

[68] Dixon, J. B. (2010). The effect of obesity on health outcomes. Molecular and cellular endocrinology, 316(2), 104-108.

[69]Verma, Avi. "Food-11." Kaggle, 5 June 2019. [Online]. Available: https://www.kaggle.com/vermaavi/food11. [Accessed: 08- Dec- 2020].

[70] "PCA-SIFT: a more distinctive representation for local image descriptors", Ieeexplore.ieee.org, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/1315206. [Accessed: 26- Sep- 2021].

[71] G. Liu, K. Liao and Y. Hui, "An improvement to the SIFT descriptor for image representation and matching", ResearchGate, 2021. [Online]. Available: https://www.researchgate.net/publication/257014990_An_improvement_to_the_SIFT_descriptor_for_image_representation_and_matching. [Accessed: 26- Sep- 2021].

[73] "PCA-SIFT: a more distinctive representation for local image descriptors", Ieeexplore.ieee.org, 2021. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1315206. [Accessed: 26- Sep- 2021].

[74] Koreascience.or.kr, 2021. [Online]. Available: https://www.koreascience.or.kr/article/CFKO201629368424723.pdf. [Accessed: 26- Sep- 2021].

[75]https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11736/117360F/Real-time-circle-detection-by-simplified-Hough-transform-on-smartphones/10.1117/12.2588773.short?SSO=1

[76] https://www.sciencedirect.com/science/article/pii/B9780124438958500106

[77]"SIFT | How To Use SIFT For Image Matching In Python", Analytics Vidhya, 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/. [Accessed: 26- Sep- 2021].

[78] "SIFT Image Features", Homepages.inf.ed.ac.uk, 2021. [Online]. Available:https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0405/MURRAY/SIFT.html. [Accessed: 26- Sep- 2021].

[79]"Feature Descriptor : SIFT (Scale Invariant Feature Transform) Part 1 : Introduction to SIFT", School of Computer Science, 2021. [Online]. Available:https://socs.binus.ac.id/2017/06/13/feature-descriptor-sift-scale-invariant-feature-transform-part-1-introduction-to-sift/. [Accessed: 26- Sep- 2021].

[80] "Introduction to SIFT(Scale Invariant Feature Transform)", Medium, 2021. [Online]. Available:https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40. [Accessed: 26- Sep- 2021].

[81] 2021. [Online]. Available: https://www.researchgate.net/publication/41890679_A_comparison_of_sift_pca-sift_and_surf. [Accessed: 26- Sep- 2021].

[82] "Bag of Visual Words in a Nutshell", Medium, 2021. [Online]. Available: https://towardsdatascience.com/bag-of-visual-words-in-a-nutshell-9ceea97ce0fb. [Accessed: 26- Sep- 2021].

[83]2021. [Online]. Available: https://www.researchgate.net/publication/228516889_Improving_Keypoint_Orientation_Assignment. [Accessed: 26- Sep- 2021].

[84]"A Step-by-Step Explanation of Principal Component Analysis (PCA)", Built In, 2021. [Online]. Available: https://builtin.com/data-science/step-step-explanation-principal-component-analysis. [Accessed: 26- Sep- 2021].

[85] "(PDF) Comparison of Feature Detection and Matching Approaches: SIFT and SURF", ResearchGate, 2021. [Online]. Available: https://www.researchgate.net/publication/314285930_Comparison_of_Feature_Detection_and_Matching_Approaches_SIFT_and_SURF. [Accessed: 26- Sep- 2021].

[86] https://link.springer.com/chapter/10.1007/11744023_32

[87] "Bag of Visual Words in a Nutshell", Medium, 2021. [Online]. Available: https://towardsdatascience.com/bag-of-visual-words-in-a-nutshell-9ceea97ce0fb. [Accessed: 26- Sep- 2021].

[88] Arxiv.org, 2021. [Online]. Available:
https://arxiv.org/ftp/arxiv/papers/1910/1910.13796.pdf. [Accessed: 26- Sep- 2021].

[89]2021. [Online]. Available:
https://www.researchgate.net/publication/331586553_Deep_Learning_vs_Traditional_Computer_Vision. [Accessed: 26- Sep- 2021].

[90]"Deep Learning Vs Traditional Computer Vision Techniques — Which Should You Choose?", Jarmos, 2021. [Online]. Available:
https://jarmos.netlify.app/posts/deep-learning-vs-traditional-techniques-a-comparison/. [Accessed: 26- Sep- 2021].

[91]"Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network | upGrad blog", upGrad blog, 2021. [Online]. Available:
https://www.upgrad.com/blog/basic-cnn-architecture/. [Accessed: 26- Sep- 2021].

[92]"A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way", Medium, 2021. [Online]. Available:
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed: 26- Sep- 2021].

[93]"What is convolutional neural network? - Definition from WhatIs.com", SearchEnterpriseAI, 2021. [Online]. Available:
https://searchenterpriseai.techtarget.com/definition/convolutional-neural-network. [Accessed: 26- Sep- 2021].

[94] "Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network | upGrad blog", upGrad blog, 2021. [Online]. Available:
https://www.upgrad.com/blog/basic-cnn-architecture/. [Accessed: 26- Sep- 2021].

[95] "Understand the architecture of CNN", Medium, 2021. [Online]. Available:
https://towardsdatascience.com/understand-the-architecture-of-cnn-90a25e244c7. [Accessed: 26- Sep- 2021].

[96] "Dense layers explained in a simple way", Medium, 2021. [Online]. Available:
https://medium.com/datathings/dense-layers-explained-in-a-simple-way-62fe1db0ed75. [Accessed: 26- Sep- 2021].

[97] "Activation Functions in Neural Networks", Medium, 2021. [Online]. Available:
https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. [Accessed: 26- Sep- 2021].

[98] 2021. [Online]. Available: https://www.mygreatlearning.com/blog/relu-activation-function/. [Accessed: 26- Sep- 2021].

[99] W. networks? and B. black, "What are the advantages of ReLU over sigmoid function in deep neural networks?", Cross Validated, 2021. [Online]. Available: https://stats.stackexchange.com/questions/126238/what-are-the-advantages-of-relu-over-sigmoid-function-in-deep-neural-networks. [Accessed: 26- Sep- 2021].

[100] Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov; 15(56):1929−1958, 2014.

[101]"Overfitting in CNNs | Learn different ways to Treat Overfitting in CNNs", Analytics Vidhya, 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/09/overfitting-in-cnn-show-to-treat-overfitting-in-convolutional-neural-networks/. [Accessed: 26- Sep- 2021].

[102]J. Brownlee, "A Gentle Introduction to Dropout for Regularizing Deep Neural Networks", Machine Learning Mastery, 2021. [Online]. Available: https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/. [Accessed: 26- Sep- 2021].

[103]J. Brownlee, "A Gentle Introduction to Dropout for Regularizing Deep Neural Networks", Machine Learning Mastery, 2021. [Online]. Available: https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/. [Accessed: 26- Sep- 2021].

[104]G. Arvindpdmn, "ImageNet", Devopedia, 2021. [Online]. Available:https://devopedia.org/imagenet. [Accessed: 26- Sep- 2021].

[105] "Deep Learning: GoogLeNet Explained", Medium, 2021. [Online]. Available:https://towardsdatascience.com/deep-learning-googlenet-explained-de8861c82765. [Accessed: 26- Sep- 2021].

[106] "CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more", Medium, 2021. [Online]. Available:https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5. [Accessed: 26- Sep- 2021].

[107] "Convolutional Neural Network Architecture", IndianTechWarrior, 2021. [Online]. Available: https://indiantechwarrior.com/convolutional-neural-network-architecture/. [Accessed: 26- Sep- 2021].

[108] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5537777/

[109] 2021. [Online]. Available: https://www.semanticscholar.org/paper/A-Method-to-Recognize-Food-using-Gist-and-SURF-Mustafa-Dhar/ed7bb78e2c379326abf30eb0084e69af177f2a17. [Accessed: 26- Sep- 2021].

[110] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision", arXiv.org, 2021. [Online]. Available: https://arxiv.org/abs/1512.00567. [Accessed: 26- Sep- 2021].

[111]"Review: Inception-v3 — 1st Runner Up (Image Classification) in ILSVRC 2015", Medium, 2021. [Online]. Available: https://sh-tsang.medium.com/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c. [Accessed: 26- Sep- 2021].

[112] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision", arXiv.org, 2021. [Online]. Available: https://arxiv.org/abs/1512.00567. [Accessed: 26- Sep- 2021].

[113] "Papers with Code - Inception-ResNet-v2 Explained", Paperswithcode.com, 2021. [Online]. Available: https://paperswithcode.com/method/inception-resnet-v2. [Accessed: 26- Sep- 2021].

[114] Arxiv.org, 2021. [Online]. Available: https://arxiv.org/pdf/1602.07261.pdf. [Accessed: 26- Sep- 2021].

[115] I. TensorFlow, "Improving Inception and Image Classification in TensorFlow", Google AI Blog, 2021. [Online]. Available: https://ai.googleblog.com/2016/08/improving-inception-and-image.html. [Accessed: 26- Sep- 2021].l

[116] "Review: Inception-v3 — 1st Runner Up (Image Classification) in ILSVRC 2015", Medium, 2021. [Online]. Available: https://sh-tsang.medium.com/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c. [Accessed: 26- Sep- 2021].

[117] I. TensorFlow, "Improving Inception and Image Classification in TensorFlow", Google AI Blog, 2021. [Online]. Available:

https://ai.googleblog.com/2016/08/improving-inception-and-image.html. [Accessed: 26- Sep- 2021].

[118] G. Arvindpdmn, "ImageNet", Devopedia, 2021. [Online]. Available: https://devopedia.org/imagenet. [Accessed: 26- Sep- 2021].

[119] "food101 | TensorFlow Datasets", TensorFlow, 2021. [Online]. Available: https://www.tensorflow.org/datasets/catalog/food101. [Accessed: 26- Sep- 2021].

[120]"Food Image Dataset", Epfl.ch, 2021. [Online]. Available: https://www.epfl.ch/labs/mmspg/downloads/food-image-datasets/. [Accessed: 26- Sep- 2021].

[121] Deep Learning With Python: Develop Deep Learning Models on Theano and tensorflow using keras, Jason Brownlee

[122] R. Verma and D. Snoopy, "Keras - Plot training, validation and test set accuracy", Stack Overflow, 2021. [Online]. Available: https://stackoverflow.com/questions/41908379/keras-plot-training-validation-and-test-set-accuracy. [Accessed: 26- Sep- 2021].

[123] "CS231n Convolutional Neural Networks for Visual Recognition", Cs231n.github.io, 2021. [Online]. Available: https://cs231n.github.io/convolutional-networks/. [Accessed: 26- Sep- 2021].

[124]2021. [Online]. Available: https://www.researchgate.net/figure/ReLU-activation-function_fig3_319235847. [Accessed: 26- Sep- 2021].

[125]Coursera, 2021. [Online]. Available: https://www.coursera.org/projects/transfer-learning-food-classification. [Accessed: 26- Sep- 2021].

[126] "Obesity and overweight", Who.int, 2021. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight. [Accessed: 26- Sep- 2021].

[127] J. Brownlee, "What is Deep Learning?", Machine Learning Mastery, 2021. [Online]. Available: https://machinelearningmastery.com/what-is-deep-learning/. [Accessed: 26- Sep- 2021].

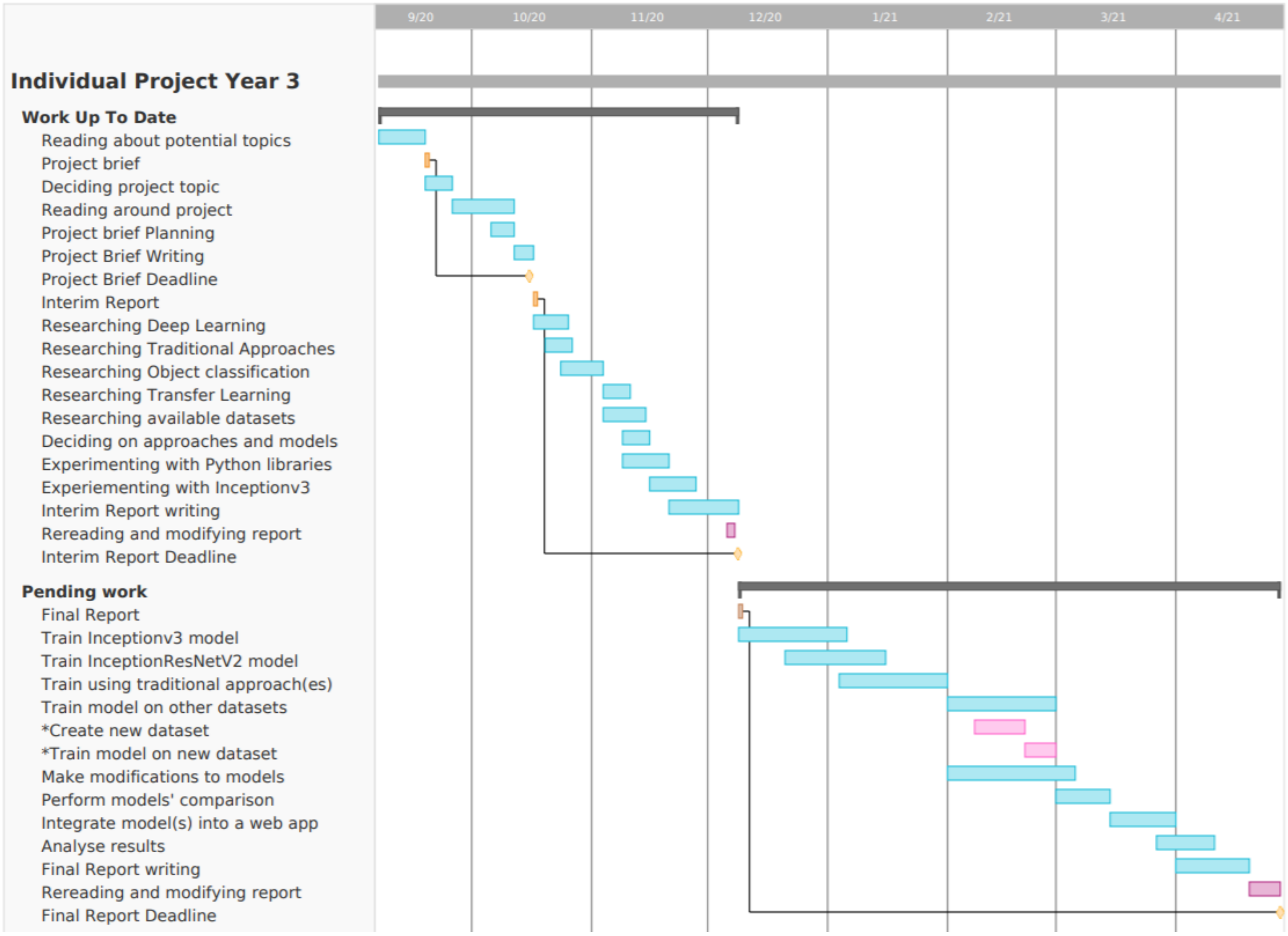
[128] Brownlee, J., 2021. A Gentle Introduction to Pooling Layers for Convolutional Neural Networks. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/> [Accessed 14 September 2021].

Chapter 7

Appendices

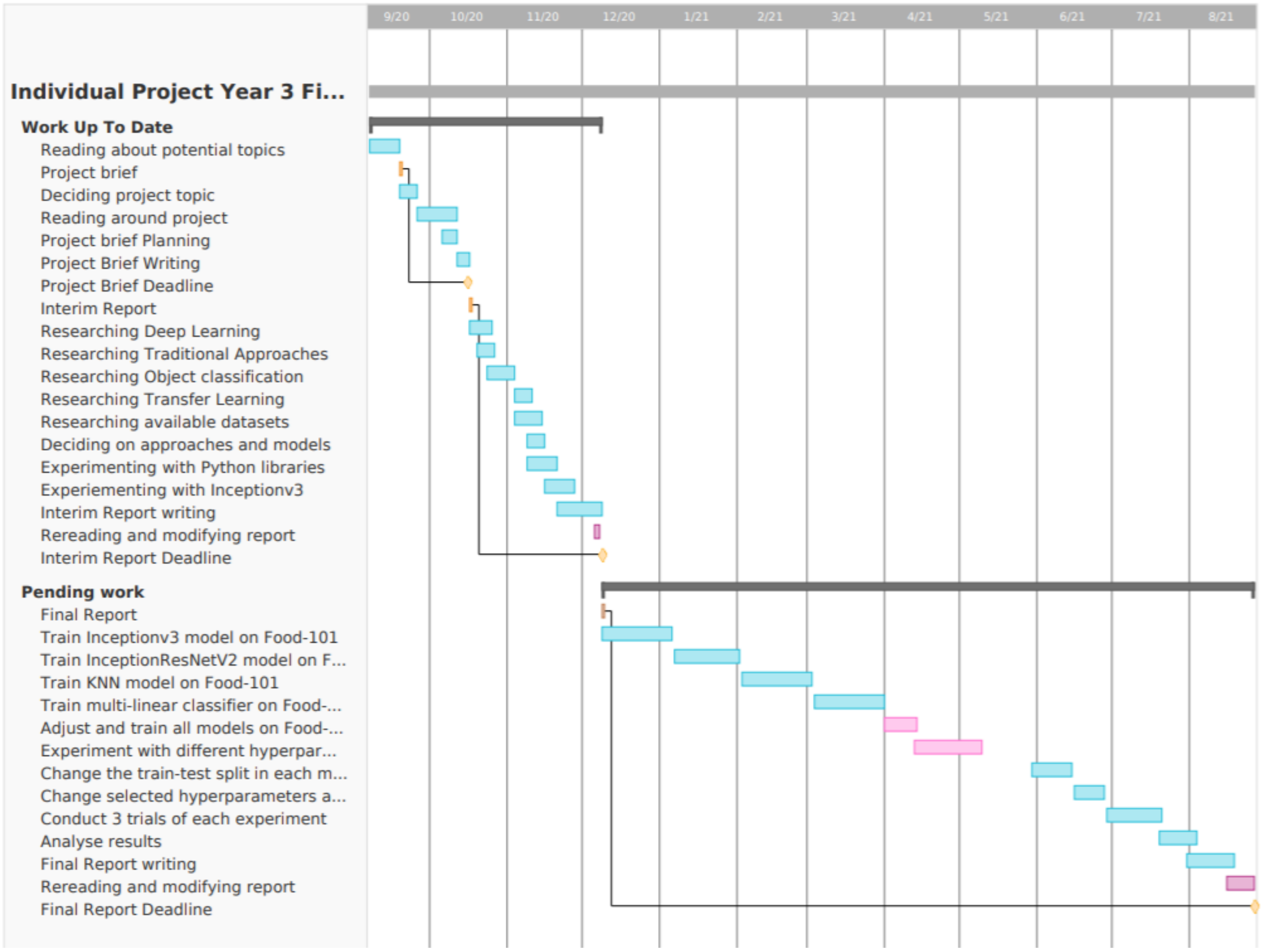
7.1 Gantt Chart

<u>Initial Gantt chart</u>



Tasks labelled with * symbols are extensions, and will only be explored if the project is up to date, and it seems feasible to work on these tasks at the time.

Final Gantt Chart



Appendix B: Risk analysis chart

7.2 Risk Assessment

Risk	Probability (1-5)	Severity (1-5)	Risk Exposure (P*S)	Mitigation Strategy
Loss of work/dataset	3	5	15	-Keep copy of written work in the cloud -Use version control to retrieve recent versions of work -Keep harddisk for work conducted locally on personal computer
Conflicting deadlines	4	3	12	-Plan and alter Gantt chart accordingly
Personal Illness	2	4	8	- Inform project supervisor and higher authority in case of Covid-19 - For not as serious situation, update Gantt chart accordingly
Lack of contact with supervisor	2	3	6	- Contact supervisor on Teams, email or organize meeting a few days in advance - Consider resources
Difficulty with implementation of models	3	3	9	- Seek help from supervisor, dedicate more time to learning and practising on less complex problems
Change in project specification	2	3	6	- Update Gantt chart accordingly -Contact project coordinator for details if in doubt

Postponed allocation of second supervisor	3	2	6	- Be prepared for the viva in advance -Keep an eye out for second supervisor allocation
Close family member(s) illness	2	3	6	- Inform supervisor and individual report coordinator - Update Gantt chart as needed
Infeasible solutions to problem	2	4	8	- Brainstorm alternate solutions -Discuss a feasible solution with supervisor -Consider alternative solutions
Project is complete too soon	1	1	1	-Work on extensions eg.new dataset/ adding features to web application
Implemented model is too time-taking and resource intensive	4	3	12	- Use GPUs in Google Colab - Connect to university’s machines locally/ in B16