Caloric Value Estimator

Joshua (Paul) Nolan
BSc (Hons)

A dissertation submitted for the degree of
Bachelor of Science in Computer Science

Supervised by *Dr, Bryan Williams*

School of Computing and Communications
Lancaster University

October, 2024

Declaration

I declare that the work presented in this dissertation is, to the best of my knowledge and belief, original and my own work. The material has not been submitted, either in whole or in part, for a degree at this, or any other university.
Name: **Joshua (Paul) Nolan**
Date: **October, 2024**

Caloric Value Estimator

Joshua (Paul) Nolan, BSc (Hons).
School of Computing and Communications, Lancaster University

A dissertation submitted for the degree of *Bachelor of Science* In Computer Science

October, 2024

Abstract

One of the biggest killers and cause of fatal diseases is obesity. Obesity is defined by a person having a body mass index greater than or equal to 30.0. The easiest and most direct solution to the problem is for people to lose weight, which is achieved by consuming fewer calories than their body expends. This is typically monitored through calorie counting or tracking which is difficult to maintain accurately, particularly where the detail of the raw ingredients is not known. The ability to determine the number of calories in a meal from an easily-captured photograph would help to tackle this issue and better enable individuals to track what they eat through the day but this is a particularly challenging problem. Limited commercial attempts at providing this solution exist, such as MyFitnessPal's 'Meal Scan' feature but this is behind a paywall, has poor user accuracy reports with no published measurements of accuracy, and the underlying methodology is not open-source or reported.

This dissertation presents a framework for a caloric value estimator which aims to estimate the caloric value of foods on a plate from a single image. This is achieved by investigating and training Deep Learning based approaches to image segmentation to find the spatial boundaries of the foods and identify them. Further, estimations of depth, scale and volume are achieved via a Food Volume Estimation model to produce an estimated volume for the foods in the image. This is then utilized in conjunction with an average food density table and a calories-content table in order to produce the predicted caloric value for the foods in the image. This paper compares, discusses and contrasts different backbones, frameworks, and overall approaches in order to evaluate the proposed solution to the challenge of estimating calories from a single image.

While the resulting segmentation model performs well for many classes of food, maintaining accuracy for a large number of distinct food types becomes challenging and the model would benefit from further investigation and refinement for more general deployment. In the case of food classes where the model performs well, IoUs of 0.4-0.6 can be achieved, which compares well to other existing food segmentation models such as FoodSAM which achieves a mean IoU of 0.46, supported by additional user input. This highlights the challenging nature of this problem and potential of the model with further development. The resulting calorific content of food examples is presented and reviewed qualitatively with

promising results observed. This work would further benefit from the development of a dataset containing the images, ground truth segmentations, 3D depth maps, volumes and calorific content of a large number of examples. With this, the model could be refined and quantitatively evaluated towards the deployment stage, with considerable potential for impact in assisting users to monitor their calorie intake more accurately and effectively, with strong potential to contribute to a reduction in obesity.

Acknowledgements

Acknowledgements.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

In recent years it's common knowledge that obesity has been a growing issue. Obesity has been on a steady rise over the last 60 years years [1] whilst showing no signs of slowing down at all. [2]Obesity is defined by a person having a BMI (Body Mass Index) equal to or greater than 30.0. BMI is a measurement of a person's weight with regards to their age, height and sex. [3] There are a number of serious health consequences associated with obesity including but most certainly not limited to:

- Type 2 diabetes
- High Blood Pressure
- Liver disease
- Kidney disease
- Several types of cancer including but not limited to:
  - Bowl Cancer
  - Breast Cancer
  - Womb Cancer

[4]
Naturally the solution to the issue is for people to lose weight. So how might someone go about doing so? It simply boils down to, almost entirely, thermodynamics meaning calories. Calories , also known and labelled on foods as kcal, are a measurement for units of energy that your body can obtain from a food. They way someone can lose weight by 'burning

fat' is to eat less calories then their body requires. [5] When a person 'burns fat' what is really happening is your fat cells are shrinking, the fat is never actually gone[6]. The reason someone loses weight when eating less than their body uses is because when you are eating less calories than needed your body will utilize its fat stores for energy causing the person to lose weight.

Calories are the unit of measure for the amount of energy a food releases when its digested in your body, but something else that is also important is macro nutrients which are what the caloric value is made up of, that being carbohydrates, fats, and proteins.
These are all very important in a person's diet as they are almost as important as how many calories someone is eating. For example if a person was to eat a very high fat diet this may not be beneficial as it can lead to poor digestion leading to gut health problems [7]. In contrast to that if a person was to eat too little fats it can very negatively effect your hormonal productions and can become very unhealthy. [8] Carbohydrates are very important as these are the body's glucose stores which are stored in the muscles and give you energy if someone was to eat too little carbs they may notice feeling rather tired and less energetic. [9] There has been a trend developing for people attempting to lose weight which is sticking to a lower carb higher protein and higher fat diet [10]. There are some potential complications that can arise from sustaining a low carb diet in the long term over many months and or years such as:

- Heart arrhythmia's
- Increased cancer risk
- Kidney damage
- Sudden death

[9] So evidently it's important for a person to be eating a healthy well balanced diet.
Finally another important aspect of macro nutrients is protein. Even if an individual isn't trying to grow any muscle or get any stronger they should still intake a sufficient amount of protein for their body weight as protein is the most satiating macro nutrient [11] meaning it can help you stay feeling fuller for longer. So clearly macro nutrients have an impact on our diet but they are not what directly causes weight loss, what directly causes a person to lose weight in spite of anything else is to be in a 'calorie deficit' meaning to intake less calories than your body expends. [5]

Your body will burn a set number of calories every day just by doing nothing. There are 4 components to how many calories you burn in a day and these are:

- NEAT

- TEF
- BMR
- Exercise

Firstly there is NEAT. You will burn calories from Non Exercise Activity Thermogenesis (NEAT) [12]. which is when calories are burnt from , as the name suggests, activities that are not part of an exercise program. This could be for example walking up and down stairs, going to the shops, getting up and down from your desk , etc. Additionally your body burns calories during exercise this could be playing a sport such as football, running, swimming, etc. Your body also burns calories through the digestion of food and this is known as the Thermic Effect of Food (TEF) [12]. And finally you burn calories via your Basal Metabolic Rate, this is your body burning calories in order to perform operations that are required for you to survive.[12]

In order for a person to lose weight we want the total of these metrics, the total daily calories you burn, to be greater than the number of calories that you intake in order to place a person into what is called a caloric deficit, which is just you expend more than you intake.

So there are 2 pretty straightforward options in order to make sure a person is in a caloric deficit. You can either increase you output, decrease your intake , or both at the same time. I believe most people would agree that mentally it is much easier to intake less calories than burn more calories, but, even if this wasn't to be true it is exponentially easier to eat less than burn the same amount or more. Many people try to outwork a bad diet but this simply isn't feasible or practical at least not in the long term of any kind. For example in the span of 10 minutes or so an individual could consume a large dominos pizza, take the Mighty Meaty. A large Mighty Meaty pizza from dominos will contain 2471 calories, according to the dominos website, in total and that's NOT including any sauces, dips, extras or stuffed crust [13]. Whilst in contrast in order to burn the same amount of calories, if you were to weight 180lbs, you would need to run 22 miles in 30 minutes! [14] For perspective The men's half-marathon, which is 13.12 miles, world record is 57 minutes and 31 seconds, set by Ugandan Jacob Kiplimo on 21 November 2021 during the Lisbon Half Marathon. [15]

I feel that this quite substantially proves my point that it is far easier to eat less than to eat a lot and attempt to out work the extra calories.
The most direct way to ensure you are in taking less calories than you are expending in order to lose weight is to track your caloric intake, also known as counting calories. People have been counting calories for decades, its not anything new. Now one reason someone might not do this could be due to the mental effort it takes and time taken to do it. It can easily become very mentally grueling and time consuming to , every time you eat something see on

the packaging how many calories are in this item and keep a count of them, not to mention attempting to track macro nutrients, these being carbs, fats, and proteins. In addition to this your faced with the glaring issue of what to do when you cant see / there is no packing for the food you are consuming.
Now in some scenarios for example when eating a meal in a restaurant, its not very feasible for most people to accurately track their calories let alone macro nutrients of the meal as not all restaurants or cafes will have the calories on their menus.[16] For some people it may be their lifestyle, for example if you're busy and working long days you aren't as likely to have the time to be weighing all your foods out let alone cooking them from scratch. The previously mention reasons lead to not many people , over the long term, measuring their calories accurately enough via weighing their foods out . If these barriers to tracking calories were removed it would, potentially, lead to a large increase in the number of people tracking their calories and therefore being far more likely to see results from a diet. This would presumably also allow them to stick to their diet for longer as they'll start to see results faster, this is where a calorie tracking app can come in.

1.1.1 Calorie Estimator

A calorie estimator is a model that can scan a plate of food and, using a given image, estimate the total caloric value of that plate of food. One of the biggest issues people have when wanting to track their caloric intake is accuracy. One study states that in one group of individuals they reportedly underestimated their caloric intake by an average of 47 percent! [17]In certain settings this can be especially true such as in a restaurant this is because in a restaurant setting if the calories are not displayed on the menu you have no sure fire way of knowing for definite the number of calories on your plate as its made by hand, and even if the calories are displayed on the menu there is no guarantee that they are 100 percent accurate. While 48 percent of British people have tried to lose weight in the last year, [18] well over one third (37percent) of the British population don't know how many calories they consume on a typical day. To track calories and macro nutrients accurately, a person should weigh their food out using a scale. This is because if you are not weighing out the food using a scale you cant know how much of that food you are actually consuming. These little errors over the course of say 4 meals throughout the day can compound very quickly. In order to be able to produce a model that can estimated the caloric value of a food there are going to be 1 major methodology that is used in order to do this called segmentation.

1.1.2 Segmentation

Image segmentation is a technique from computer vision, a sub-specialty of artificial intelligence. Segmentation involves breaking an image down into segments, or objects. For

example we might outline a dog in a photo, or a person, or a bike, etc. Within segmentation we have semantic segmentation, this is where different objects are outlined and also labelled, so it will distinguish between a person and a car but will also label them as a person and a car.

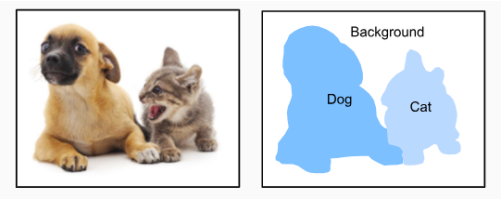


Figure 1.1: Example image showing a typical mask output for a segmentation model of a dog and a cat. [19]

[19]

As previously mentioned there is segmentation and semantic segmentation , the models that are going to be discussed will be both segmentation (Where objects or subjects are outlined) and semantic segmentation (where the outlined objects are classified).

1.1.3 CNN

Alongside segmentation in order to implement this we would make use of a Neural Network more specifically a CNN (Convolutional Neural Network) [20]. A Neural network is a deep learning model that attempts to mimic the human brain by using 'nodes' also known as 'neurons' interconnected which allows the model to learn from data and adapt to then be able to correctly predict or recognise certain features from future data it is passed. A convolutional neural network is a neural network that , as the name suggests, contains convolutional layers which is where features are extracted, and pooling layers, which down sample the output from the convolution then this process is repeated many times in a CNN to eventually produce a map of features which is then passed to the final layer where the classification of the objects segmented in the mask is done to produce a final output.

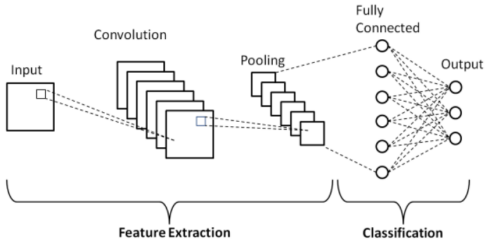


Figure 1.2: Image showing basic version of a CNN. [21]

[21]

1.1.3.1 Pooling

Pooling is something that is very common in CNN models, Pooling layers are used down-sample the size of the feature map that the convolutional layer will have passed to it. It achieves this by reducing the resolution of the feature maps which reduces the number of parameters for the model to have to learn. Pooling layers work by summarising smaller areas of a larger feature map generated by a convolution layer to produced a summarised version to then be passed deeper and further into the model. Pooling functions will have a stride this means after it has pooled a certain area how far in the x and y plane should the filter / pooling function be moved along to begin the next pooling on the next region of the image.

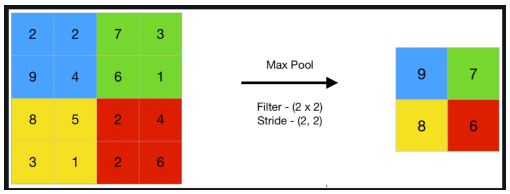


Figure 1.3: Image showing in an easily visualised way how pooling works, in this case Max Pooling with a filter size 2x2 and a stride of 2,2 meaning moves 2 pixels across and then 2 pixels down when it reaches the end horizontally [22]

1.1.3.2 Classification

Classification is the last section of the CNN before producing the final output and this is where the model determines which class it believes each segmented object in the image belongs to. The way this is done is by implementing a confidence score. For each segmentation the model will have produced a confidence score, usually between 0 and 1. 1 meaning its 100% certain that the segmented object is of that class and 0 meaning its 100% sure that it is not part of that class and whichever class has the highest confidence score for that segmented object in the image is the class the model will assign to that segmentation. [23]

1.1.3.3 Evaluation

When creating a model to solve this problem the model must be evaluated effectively upon completion in order to determine how effective this model is at solving the problem. Metrics such as IoU [24] will be important and Dice to determine what is the most effective and accurate model for the segmentation. Its also important to look at the calories that the model produces and the macro nutrients as this is the goal of the model and comparing these to the calories labelled on the package and weighting the same food out by hand to test the accuracy of the number produced for the calories of that food.

1.1.3.4 DICE and IoU

During testing a model there are 2 variables that we use to track the progress and improvement of the model. DICE indicates to use the error rate at each epoch. An epoch[25] just refers to 1 iteration of a model, when the model is training it is underoging many epochs as after each one is where the weights are adjusted for each node according go the loss function ran at the end of each epoch. An example of a loss function is the Sørensen–Loss function is defined as:

$$L = 1 - D(A, B) \tag{1.1}$$

[26] It represents the difference in 2 masks, A and B in this case referring to the similarly between the ground truth mask and the mask produced by the segmentation model and here D is the Dice function.

$$DICE = \frac{2 \times |A \cap B|}{|A| + |B|} \tag{1.2}$$

[27] DICE scores range from 0 to 1. 1 indicating a pixel perfect match from the produced mask and the ground truth and 0 meaning no matching pixels whatsoever the larger the score the larger the overlap between the 2. The other variable we use to track the performance and accuracy of our model is IoU. IoU stands for Intersection Over Union and is defined as:

$$IoU = \frac{A \cap B}{A \cup B} \tag{1.3}$$

[24] Its similar to DICE in the sense that it also measures how much overlap there is between the segmented region produced by the model and the ground truth. The ground truth(s) are a set of images where there is 1 for each image in a data set. They are masks created by a person in order to show what the correct segmentation of an image is with each object of a different class in the image having a corresponding shade / colour dictated by what its been classified as. Its calculated by taking the area of the overlap divided by the Area of union which is the total size of the two segments combined. Effectively it is , as a percent, what amount of the boxes overlap each other.

1.1.4 Depth Estimation

One Challenge we must overcome within this project is estimating depth and in turn the volume of the food(s) in the image. Of course in order to be able to estimate the caloric value of a food we need to know how much / what volume of that food we can see in order to estimate its weight. It should be noted that in order to accurately estimate depth from a single point of view image is very very difficult. As the model has to take into account the plane in which the photo was taken, by plane its mean the position of the camera taking the photo relative to the subject as this will effect how much of the food(s) can be seen in

the image and although being one of the most challenging methods, it is the most widely applicable.

One depth perception technique we may consider using is Lidar (Light Detection and Ranging) is a remote sensing method that uses light pulses to measure distance from it to the subject. [28] Typically it works by sending out pulses of light which then will bounce off of the subject and return back to the sensor, lidar then calculates how long it took for the light to return back to the sensor and then knows how far the object is away from its sensor. This process is repeated millions of times per second and this is how it can construct a 3rd representation of the area it has covered. The issue with lidar within this project is that if this methodology was to be published and eventually deployed onto for example a mobile phone. there is very few phone models it would be compatible with and we want something that is very broadly accessible. Not all phones would contain lidar sensors for example apple's iphone smartphones do have them but what about android phones or more budget options.

Another challenge that may arise when attempting to use lidar as the depth estimation for this project would be the fact we have to make some assumptions within the depth estimation model. Firstly I would have to make an assumption about the width of the plate that the food is placed upon. Lidar alone can not estimate the width of the area its looking like. I could consider firstly use as said, an estimated width for the plate the food is on and I could also use the segmentation map to then estimate the width of each class of food.
Another accommodation that may have to be made might be in a real world app is to prompt a user to take all photo inputs on the same plane preferably from a top down view in order to have consistency across the model.

1.2 Project Aims and Objectives

The goal of the project is to develop a methodology to estimate the number of calories on a plate of food by breaking down the plate into each foods caloric value and also macro nutrients. Thus allowing for a much easier and time efficient way for a person to track their caloric intake. We aim to have a model that has a realistic accuracy whilst maintaining a relatively low enough overhead so much so that we are not sacrificing one for the other. We can achieve this goal by breaking it down broadly into the following 2 objectives:

1. Firstly we need to be able to semantically segment the food we are viewing in the image, by this we mean being able to identify the boundaries of these foods thus being able to separate them all out from one another and so being able to estimate what it is on the plate.
2. Secondly we then need way to be able to, using the boundaries just outlined, estimate the total weight of the food that is there and of course doing this through just an image on a

screen can be tricky so we can are going to break this down into 2 sections:

Section A will be to Firstly use the image to estimate the total volume of each food that is on the plate this is a very challenging task. The main task making this very difficult is the fact that we are trying to estimate volume from a single-point of view, the photo the model is estimating from could be at an awful angle for example a pile of mash potato from a photo that is quite low down side on or for example a photo that is top down u don't really have much of a way of estimating the depth as its efficiently a 2D image at that point. There are a few approaches we could consider for example we may consider an approach called 'lidar'. Lidar uses a light beam it sends out and based off of how long it takes for this to return back to the sensor that it was sent from the lidar model can estimate a depth map of what is placed in front of it this shall be explained in more detail later on. Another approach is to use a pre-trained volume estimation model ideally specifically designed for volume estimation of food if possible.

Section B will be then using this estimated volume for the food(s) in conjunction with a average food density table [29] that will contain average densities for the food(s) we have estimated the volume for to produce an estimated weight and finally from this estimated weight we can utilise a table of the food(s) caloric and macro nutrient values in order to estimate for the food(s) in the given photo a caloric and macro nutrient value. [30]

1.3 Motivation

The main driver of motivation for the project would be due to the on going obesity issue in not just the United Kingdom but world wide. A person is classed as obese when they have a BMI (Body Mass Index) of 30 or higher and they are classes as morbidly obese when they have a BMI of or greater than 40.

BMI is a metric that is used to estimate if you are a healthy weight by comparing your weight and sex against your height.[31] Although BMI isn't perfect and does have some flaws such as not accounting for varying bone density and if a person has a lot of muscle tissue on their frame, for the broader population its a good guideline to roughly track if you are a healthy weight or not.

Although this is something most people are aware of, not everyone makes the decisions to do something about it or in some cases can not. In 2020 across the United States of America 33.1% of Men were obese while 33.9% of women were obese (having a BMI over 30)[32] . Potentially even more alarming is childhood obesity rates, in 2011-2012 16.9% of children [33] were obese with almost 1 third being overweight or obese (31.8 percent). This is clearly a massive issue. The number of related disease and issues that can arise from being severely overweight or obese is never ending. For example:

- Type 2 diabetes
- Numerous types of cancer
- Stroke

[3] Another potential motivator for this methodology could be for diabetic and pre-diabetic people , when someone has diabetes they need to have enough or not too much sugar depending on whether they have type 1 or type 2 diabetes.[34] If this methodology was deployed into a application for users this would make tracking sugar intake much easier allowing users who are diabetic to ensure they are everyday achieving the correct sugar intake needed for their body.

1.3.1 Losing Weight

As previously mentioned the way someone 'loses weight' which is really just the shrinking of fat cells which in turn reduces your net body weight, is by being a calorie , meaning to burn more calories than your body intakes in a day which , due to thermodynamics will cause you fat cells to shrink and in turn lose weight. Now the most straight forward, obvious, and efficient way to do this is to count calories, tracking the number of calories that you intake in a given day to find a number of calories to intake daily that is lower than your daily expenditure.

Your **daily expenditure** of calories is made up of as we already mentioned 4 attributes. [12] BMR , known as your resting metabolic rate, NEAT (Non-exercise activity thermogenesis), exercise such as walking or running, and TEF known as the thermic effect of food. It is very very difficult to track and or measure these metrics yourself at home day to day. By far the most practical way and easiest way to determine your estimate daily caloric expenditure is to use something called a TDE calculator. A TDE calculator (TDE standing for Total Daily Expenditure) website such as tdeecalculator.net [35] is something that uses your weight,age,height, and activity level to estimate your total daily expenditure, this is a great starting point that can be used to estimate your expenditure. The way you would use this to go about finding your body's daily expenditure would be to track your calorie as accurately as possible by weighing all your food out on a scale before consuming and or cooking the foods and then every morning immediately after going to the toilet but before eating your first meal tracking your weight on some scales in the same clothes every morning such as your boxers or pajamas in order to allow for fairness and consistency morning to morning when weighing in. If you track your calories and intake the same number as calories as your estimated daily expenditure over the course of a week or 2 you should see your weight maintain. If u gained for example 0.5kg you can try eating maybe 200 calories less the next week and visa versa if you lost 0.5kg, this method of experimentation and trial error is the

most effective way to learn your body and how many calories your burning in a day. It's also important to consider exercise so for example your iphone or if you have an apple watch these devices attempt to using their built in accelerometer track your daily step count and keeping a track on this and aiming to increase it to a certain goal number can be an easy way to burn more calories.[36]

One potential barrier to entry to track calories is the effort required to keep it up everyday or even if you do mange to another major barrier to entry in doing this is that it's infeasible sometimes impossible to track the calories of a meal when eating at a restaurant unless stated on the menu and not all restaurants will for example when in a smaller pub its unlikely they'll show calories on their menu. [16] There are already some current approaches notable in another app named my fitness pal who have attempted to tackle this issue with the use of an AI model to interpret an image of food(s) in order to estimate caloric and macro nutrient values and one of the largest in this area is MyFitnessPal with over 200million users worldwide. [37] When looking into how their application works we find that according to the my fitness pal website on their FAQ page under a section labelled 'How does the technology work' they state:
"MyFitnessPal uses Machine Learning and Computer Vision to detect and recognize foods using images from your smartphone camera. Leveraging models built and run by MyFitnessPal that are trained on millions of images, it analyzes the images and suggests verified foods from our vast food database." [38]
This is one of the biggest issues with current approaches is that because they are products being sold by companies as appose to research being done they are not open source and trying to find the details of what architecture and methodologies these applications are using can prove very difficult. Due to it being a product for profit and so it isn't open source, there is no measure or way of knowing what measure of accuracy they have used in their application in order to be able to compare against .
So since the code is not re viewable as of course these companies want to protect their product, this makes it harder for research as we are not able to see what the current approaches are in this area in order to be able to potentially build upon the existing models to make improvements. Another issue when some applications in this area such as My Fitness Pal is that they are behind a paywall. The My Fitness Pal offers an AI implemented solution similar to the idea presented here but it is locked behind a paywall meaning we cant use it for experimentation to compare with but it also means that from the perspective of the users this is another barrier to entry to use their product.

This approach of using AI models for tracking calories is something that when applied in the manner proposed could help somewhat tackle the obesity issue through AI. If someone had this methodology on their mobile phone in a manner that is easily accessible and fast, it would remove a good number of the barriers to entry to tracking calories that stopped

someone weighing all their food out on a scale such as time, effort, money. Therefore this would allow someone to have a healthier more balanced diet to better take care of their health.

1.4 SCOPE AND LIMITATIONS

When beginning this project there are some limitations that are clear are either impossible or unfeasible to overcome during this project, these will , in different ways, limit the models potential currently and until they're overcome prove problematic in given scenarios. One Limitation that, even before beginning any coding or form of testing, I am aware of is something I am going to refer to as the "Chicken and Rice" problem. The problem is as follows; Assume you had a bowl containing and mixture of chicken and rice. There is no possible way for the model, from an image of the bowl, to determine how much chicken and rice is in bowl within the photo. Not just this but also there is the issue of whether or not there is other items of food within the bowl that are simply buried below by chance that are not visible from the surface. In addition to this the problem may become particularly tricky if the photo is from a perfectly top down view. If the image was to be taken from a

Figure 1.4: Example showing the difficulty in perceiving depth from a top down image [39]

slight angle further away from the food the model could at the very least assume the ratio of rice and chicken it sees on the surface is repeated on the way down and because of the slight angle the model may be able to estimate the volume of the bowl and therefore how much chicken and rice is inside the bowl. Something that we could use to tackle this potential issue is with the use of lidar. [28] For top down images lidar would be able to take the difference between the surface around the bowl (the surface the bowl is on) and the top layer

of food in the bowl in order to estimate the depth of the bowl, but this still has issues. For example it cant not detect the shape of the bowl and so wont know how much the bowl curves as it moves closer to the table effecting the volume of food the bowl is able to hold. This is an issue that is fundamentally impossible to solve currently with a single photo model.

Another issue that the model wont be able to overcome without some kind of user input would be how the food was cooked. Although this may not seem a massive problem it can effect the calories massively. One tablespoon of butter contains 102 calories and 11.5grams of fat with 7.3 grams of these being saturates while 1 tablespoon of olive oil contains 119 calories and 13.5 grams of fat. Although this 13.5 grams of fat only contains 1.8 grams of butter. [40]
The calorie difference can increase when we are using multiple tablespoons and regardless of how much we use the different in saturated fats is notable.
The issue arises in that there is little to no visual difference that the model could use to detect whether some foods were cooked in butter vs olive oil thus impairing its ability to accurately track calories. This may be able to be mitigated by having some form of user input in a fully deployed version of this model in which after the model completes it can ask the user a short set of questions for example "Was this food cooked in oil, butter, none, or other (please specify)" thus allowing the model to have a more accurate representation of how the food was prepared.

One further limitation that may arise during the development of this model that is infeasible rather than impossible to solve is the challenge of understanding the size and dimensions of the plate to then help us in the depth perception of the food on the plate. This is because if the image is taken at an angle as appose to a top down view, then the model will have to take into account the plain that the photo was taken on in order to better estimate volume as it cant treat a photo taken at an angle as a 2D view as this would give an inaccurate measurement for the volume of food in the image. This is something that may be considered to be outside the scope of this project as it would require more research and work in order to perfect to be able to produce a well-rounded model with accurate values. This would be most effective when done as a 3D representation which is a topic that is undergoing current research and something that would very very difficult especially when considering from the standpoint of just using 1 photo taken as the input image.

Another possible solution would be to create a classifier specifically for the problem meaning to have a classifier that would recognise the dimensions of the bowl / plate and be able to map it to the plate or bowl it believes is the closest to what is shown in the input image, this would still not be 100% accurate as we are just classifying to the item we believe it to be the closest to. This is not something that is feasible during this project as during the time frame that was given to complete this it was considered more important to focus on the

segmentation part of this model since the segmentation of the image is far more fundamental to the success of the model overall. This is because in spite of a perfectly accurate depth perception for the food your estimation for the calories would be widely wrong if for example chicken breast, which is around 106 calories per 100g, was thought to be sausages, which are in contrast 301 calories per 100g. [41] [42]
One final limitation that is rather glaring due to some assumptions that have to be made for this project is the case of what if a photo is take where there is no plate or table for example someone holding a burrito up in front of the camera? Here the model may struggle as there is no close up point of reference such as a table for the model to be able to contrast the depth of the burrito against in order to estimate volume and thus calories. This is an area that could require further research in something such as an advanced 3D model reconstructive program although this is current cutting edge research and something that is very very hard to perform accurately.

Chapter 2

Background

2.1 Background and Related Work

In order to estimate the caloric value of the food on a plate using an image taken with a smartphone we are going to use different AI components, this section shall discuss different architectures and approaches to implementing segmentation into a model that will work for our problem whilst being effective and efficient.

Image segmentation in computer vision is when an image is inputted and then partitioned into different sections or objects, for example it can distinguish a cat sat next to a dog as 2 different things/objects but they are not identified they are simply recognised as separate because the model can outline the border of what region of the image is an object.
This alone can be very useful as without this we wouldn't be able to label any objects.
Within Image Segmentation we have different architectures we can use. One of the earliest models of segmentation would be Threshholding [43], this is where each subject is put through a logistic regression model and is given a score, between 0 and 1, then it is determined which class it should be assigned to based off of this score.
For example a score of or over 0.5 may be assigned to class 2 and a score under 0.5 may be assigned to class 1 this is also by nature doing the job of classification as well as it is classifying the objects into different classes.
When building the model it will be implementing a Neural Network as these are best suited towards learning complex patters and or problems such as ours and can learn quickly from the data it is given more specifically a Convolutional Nueral Network (CNN) .

2.1.1 Neural Networks

Neural networks are a type of machine learning model that contain interconnected neurons or nodes that take data and learn from it. Data is passaed to the input layer which is then passed to the nodes in the hidden layer which is then passed to the output layer. Inside of

the hidden layer is the nodes where the learning is done one way in which the nodes 'learn' or are updated to better suit the problem its given is via using Back Propagation:

2.1.1.1 Back propagation

Back propagating is something that is very common in many neural networks and it is where we use the error rate or loss obtained in the previous epoch to adjusts the nodes / neurons weights in order to improve the model.
[44] Inside each layer is where the nodes apply linear functions using weights to determine an output. The weights are the weights of each neural connection and determines how strong the connection is between it and the node at the end of the corresponding connection. Once this has been done some neural networks will contain next an activation function.

2.1.2 Activation Function

An activation function will take the output of the linear function and alter it before its passed to the output layer, this is done as not all data is linear and allows the model to be more robust and adaptive to real data. One use of activation functions is if you had 2 convolutions , A, B, and an image X.
Applying convolution A to image X to give AX then applying the second convolution B to give ABX would be the same as applying the convolution A to convolution B to give AB then applying that AB convolution to X to give again ABX. An activation function is used after each convolution is applied to the image so they can have their desired impact on the image.

2.1.2.1 ReLU

One example of an activation function might be RELU. RELU (Also known as Rectified Linear Unit Activation Function) is a piecewise non-linear function that will convert any negative values into a value of 0, this also means 0 is mapped to 0 and anything greater than 0 will be mapped to the value its self and will remain unchanged.
This threshold method at 0 and below introduces non linearity into the model and also when multiple RELU functions are stacked , via having multiple layers with neurons passing each RELU function's output to the next neuron and so on this introduces even more non-linearity into the model as with each layer more and more non-positive values are eliminated. [45] The reason an activation function such as RELU is required is because if an image A is to undergo 2 convolutions, B and C, this is equivalent to C convolving B to produce 1 single convolution D then the image A having the convolution D applied to it.

Within Neural Networks the model will be using specific type of Neural Network that is well suited towards image processing and computer vision and these are known as Convolutional Neural Networks:

2.1.3 Convolutional Neural Networks (CNN)

Convolutional Neural Networks are a form of deep leaning neural network that are most commonly used for pattern recognition and also image segmentation. They are primarily made up of convolutional layers along side pooling layers. An input image is passed to the network and its first passed to a convolutional layer.

2.1.3.1 Convolutional layer

This is where kernels, which can be thought of as a filter or mask, of a size smaller than the image are applied onto a section of the image and shifted along until the entire image has been scanned over by this smaller size filter called a kernel which allows us to extract features from the image. As we move the kernel along we get an output each time which are passed to the pooling layers and we combine the output of these pooling layers together to get our output which is known as a feature map.

In order to be able to estimate the boundaries of a subject we typically use some form of edge detection filters. An edge detection filter is , simply put, a mulit-step algorithm that is used to detect the edges of objects in an image using changes in intensity values.
Inside of this process there is something called a kernel. A kernel is a filter, so it may be a 4x4 grid with each square in the grid containing a value to be used as a multiplier to the value it sits above when the kernel is shuffled along or placed on top of an image. Kernels have a stride, a stride refers to how far the kernel is moved along at each iteration for example a stride of 3,3 means the kernel moves along 3 pixels every time until it reaches the end of the row and then moves 3 pixels down and resets to continue left to right until it has covered the entire image. Each time the kernel covers a for example 3x3 section the output of whatever we perform with our filter is stored in a compressed version of the image data in our output matrix. We have different types of edge detector filters we could apply to the image we are considering for example Sobel Operator or Prewitt Operator. [46]
The Sobel filter is displayed below:

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 2.1: The Sobel Filter [47]

Here we can see Gx which is used when deriving in the x direction and similarly Gy for when deriving in the y direction. The Prewitt filter can also be seen below and in the same fashion as the Sobel Filter u can see here the filter for the X and Y derivatives

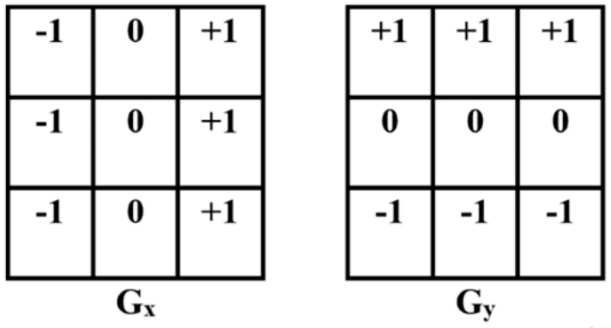


Figure 2.2: Prewitt Filter [48]

Depending on the scenario you are using these filters in may determine which you choose to use, for example the Sobel filter puts more of an emphasis on the centre pixels having the +2 and -2 values on the squares closest to the centre whereas the Prewitt filter is a more broad filter treating all sections of the filter equally

Noisy images may disrupt these filters though so taking this into consideration you may want to give some thought to applying a smoothing filter to your image before performing any edge detection. Some examples of smoothing filters may include Gaussian filter or Average filter.

After the input image has been passed through the convolutional process it is then passed to that same layers pooling function to be summarised and then passed to the next layer.

2.1.3.2 Pooling

Pooling is something that is very common in CNN , Pooling layers are used in order to reduce / down-sample the size of the feature map that the convolutional layer will have passed to it and are also used to save on computation time meaning the model is faster. It achieves this by reducing the dimensions of feature maps which reduces the number of parameters for the model to have to learn. Pooling layers work by summarising smaller areas of a larger feature map generated by a convolution layer this summarised version of the feature map containing now summarised features is then passed along the net work instead of the more complex detailed version. As a side effect of this it also makes the model more robust as it will mitigate the impact of small anomaly's on our model as much as possible.

There are different kinds of pooling layers and different methods we can apply. An example of one of these would be Max Pooling. Max pooling is a method in we take the largest value from a region and take that value to represent that entire region in our condensed

representation, therefore we take only the most prominent features from the larger more detailed map in our condensed summarised version.

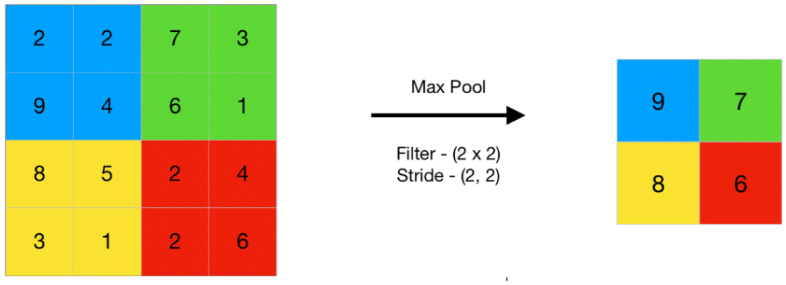


Figure 2.3: Example of max pooling function[22]

Another example of a pooling function might be Average pooling. As the name suggests here the idea is to take an average of all the values in a local area and then takes the average of those values to represent that segment in the condensed smaller version, by taking the average value of a segment any outliers hold much less if any weight in the model and so the overall and most common features are represented in our condensed version.
Within different pooling techniques we can adjust the size of the how far we move the filter along in each iteration and this is known as the 'stride'. So for example if we had a 3x3 filter a stride of 1 would mean at each iteration its only shifted over and or down by 1 pixel covering pixels multiple times where as a stride of 3 would mean we jump 3 across and or down from the starting point upon each iteration.

2.1.4 Confidence Scores

A CNN model needs to be able to asses its own predictions of the images and what classes it believes to be in the image or not in the image and One way the model is able to do this is by making use of a metric called a 'Confidence Score'. A Confidence score is a metric that is assigned to every segmentation output, typically valued between 0 and 1, and indicates how strongly the model believes the that its prediction is correct. For an example a model may use parameters such as; 0 being there is no chance it believes that this image contains that specific class whereas a confidence score of 0.7 would suggest its 70 percent confident and if this was to be over the thresh-hold of for example 0.6 it would then satisfy the requirement to exceed the thresh-hold and so would classify those pixels as that class.

2.1.5 Background History of Segmentation

The first early stages of segmentation was started in the 1980s using thresh-holding to begin with. In short the basic principle of thresh-holding is to take an image and be able to separate the image into a foreground and background. Thresholding produces a binary image (every pixel represented by a 1 or 0) where it uses the produced intensity values to determine what the seperate a particular pixel into dependet on whether the intensity value of that pixel is equal to, below, or above the given intensity threshold value.Expanding upon thresh-holding we have global and variable thresh-holding, Global being where a set threshold T is applied across the entire image and variable being where T varies across the image for different ranges of intensity values. The way we can determine and choose the best threshold for our value is to try different values and look at their false positives and false negatives at these values to compare which is the most appropriate to our problem. To explain false negatives and false positives I shall use the example of scanning for cancerous tumors. False negatives in this case would be when we produce a result saying there is no tumours, when in reality there is. And false positives are when we say there is a tumour when in reality there is not.

Something that was used early on in the late 80s moving into the early 90s was a functional that was called the Mumford-Shah functional. The Mumford-Shah functional was first published in a paper named "Optimal Approximations by Piecewise Smooth Functions and Associated Variational Problems" which was published in 1989 by DAVID MUMFORD from Harvard University alongside JAYANT SHAH from Northeastern University. [49]
The mumford-shah functional is defined as:

$$E[J,B] = \alpha \int_D (I(\vec{x}) - J(\vec{x}))^2 \, \mathrm{d}\vec{x} + \beta \int_{D/B} \vec{\nabla} J(\vec{x}) \cdot \vec{\nabla} J(\vec{x}) \, \mathrm{d}\vec{x} + \gamma \int_B ds$$

Figure 2.4: Mumford-Shah Functional

[49] In this functional any input image is modelled as a peicewise-smooth function. The functional is designed to smooth homogeneous regions in the image and at the same time enhance edges, the function does this by penalizing the distance that is between the model and the input image, the lack of smoothness within a model and the length of the boundaries of the sub regions.

In the year 2001 a paper called "'Active Contours Without Edges" by Tony F. Chan and Luminita A. Vese ' was published outlining an image segmentation model that aims to use contours / edges for image detection this was later to become known as the "Chan Vese" model [50]. Its mainly designed for the challenge of segmenting objects that don't have clearly defined boundaries. The model is based on the Mumford-Shah functional and is commonly used in medicine for segmentation of things such as the brain or the heart. It essentially considers all possible positions of a curve drawn around an object and only accepts the position of the curve that satisfies or best satisfies the equation below:

$$J_1(C) = \alpha \int_0^1 |C'(s)|^2 \, ds + \beta \int_0^1 |C''(s)| \, ds - \lambda \int_0^1 |\nabla u_0(C(s))|^2 \, ds.$$

Figure 2.5: Chan Vese equation from their model [50]

The variance inside the contour is minimized and the the variance outside the contour is also minimized and the 3rd term ensures that the length of the contour C, should be as small as possible whilst still satisfying the first 2 conditions.

Moving into the 2010s Machine learning was a more prominent and effective technique for image segmentation. Machine learning is a form of AI that uses algorithms and statistical models to learn and attempt to make predictions about future data based of the data it has been passed in the past in order to make future predictions. Here we are using predetermined masks which are known as ground truths and corresponding images in order to teach the system how to recognise different categories of objects. An example of how a system using this may work is the method stated above thresh-holding for image recognition. As its not using a neural network it is not classified as Deep learning and is Machine learning. These types of models are useful but don't have the capability to learn on their own without human input in the same ways that a Deep Learning model can.

Moving to 2015 and onwards to present day, Deep learning has become a much more prominent, effective and efficient technique for building image segmentation models. Deep learning, as apposed to machine learning is when instead of just having an algorithm run by itself over a piece of data, we are using a neural network which enables the model to actually learn from multiple iterations or 'epochs' of training on the same data set effectively allowing the model to learn almost like a human might. Neural networks also learn more as they interpret more data thus allowing them to perform better, than machine learning models that don't use neural networks, at learning from highly complex larger data sets therefore being better suited to more real world problems.

There are 2 very popular architectures that would both be classified as Deep Learning Models and will be considered for use our methodology in the project: These are UNET and UNET++.
I'll begin by first explaining UNET. UNET is a CNN architecture that has a U shape to it. It contains 2 paths both made up of multiple layers. The first path is used to encode the input image and encode the features that are detected in the input image and reduce the resolution of the image. The other path which comes 2nd to the previous path is where the model decodes the encoded low resolution images generating by the first path in order to , along side using skip connections, generate a segmentation map of the original input image.

SKIP CONNECTIONS Skip connections are used in CNNs such as UNET to try and save computation time and or cost thus making them more efficient. They are connections between an encoding layer and its corresponding decoding layer on the other side of the network. Doing this allows the decoding layer to have more information about what is is trying to decode and the mask its trying to produce.

2.1.5.1 UNET

UNET [51] is a fully convolutional neural network that was developed in 2015 for the purpose of image segmentation superficially it was originally designed for biomedical image segmentation.
At each layer in the UNET architecture there is a convolution layer performing a convoluion and then followed by an activation function such as RELU before its then passed to the next layer via pooling, this could be max pooling or avg pooling, this is repeated at each layer untill it reaches the bottom of the model and then it performs a convolution but this time instead of perform a pooling function down to the next layer we upscale the image , doing the inverse of a convolution, where its passed to the decoder side of the network where similarly to the encoder at each layer it performs convolutions and then upscales the output before its passed to the next layer above it creating a U shaped architecture, hence the name. Below is a figure showing the general U shaped architecture for a UNET Model.

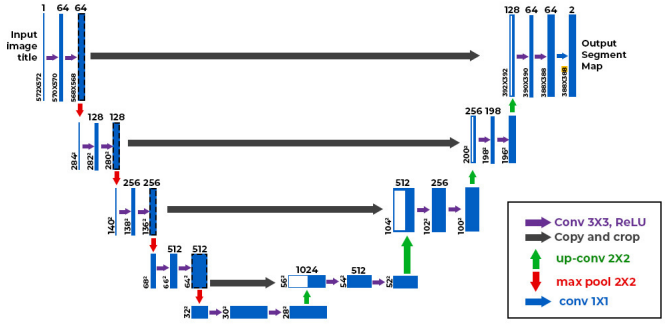


Figure 2.6: Image visualising the UNET architecture [52]

2.1.5.2 UNET++

UNET++ [53] is an architecture that is built upon and improves upon UNET. UNET++ , which is also known as Nested UNET, introduces 2 new modifications to the original UNET these being **skip connections** and **deep supervision**. Inside UNET we have the previously mentioned encoder and decoder sides to the architecture with the layers of convolutions and

pooling functions. The problem here lies in the fact that there can be a semantic gap meaning that the feature maps coming from the encoder are at a lower level than the level of the decoder trying to decode said feature map, this in turn may cause the decoder to struggle to reconstruct the encoded feature map with complete accuracy. UNET++ introduces skip connections to solve this issue. The skip connections allow the decoder to make use of low-level and high-level feauters from different layers from the encoder. UNET originally does have skip connections between the encoder and decoder layers but UNET++ uses **nested** skip connections meaning between each encoder and decoder layer we have multiple skip connections. In each of these nest skip connections the image from the encoder side is up-scaled to match the convolution to the decoder we are connecting to as seen in the diagram below:

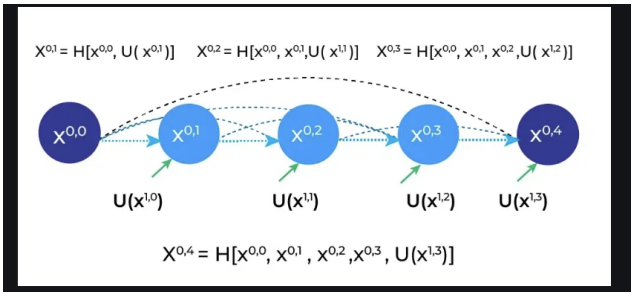


Figure 2.7: Figure 1.1 Showing nested skip connections [54]

Unet++ also implements **deep supervision** which helps to improve the models performance by supervising we are comparing it with our ground truths are so are more easily able to see when and or where any errors are made and potentially correct them or use these to further improve our model. This involves adding in an additional loss function at each intermediate layer. This also helps to solve the **vanishing gradient problem**

We start out with a data set of images of a variety of different subjects in each image. Each of these images has a mask image paired with it these may be manually by hand segmented. These preset masks are known as the ground truth of the images and they are used later on to compare the output of our model against it as we know the ground truth is correct for the given image and so can be used as a benchmark. The model is trained on this dataset to be able to understand for example what an apple looks like so that when a new image of an apple is given to it, it can recognise it is an image of the category apple.

In order for the model to extract features from a new image, once we have our data set ready for use, the encoder receives our input image and extracts features from the image using multiple convolutions layers along side max pooling layers that extract the features. It then up samples these features and concatenates the encoders features with the decoders features via paths that connect the encoder to the decoder at each level. The final layer is then what produces the output giving us our segmentation mask of the original image. We

can then calculate our loss via comparing the output mask to a ground-truth mask allowing us to determine the accuracy and efficiency of our model

2.1.5.3 PSPNET

Another example of a modern day CNN architecture would be Pyramid Scene Parsing Network (PSPNET): [55]
This is based on semantic segmentation and is aimed at identifying specific parts of an image where the goal is to assign each pixel in the image a category label. Scene parsing provides complete understanding of the scene. It predicts the label, location, as well as shape for each element. Context is very important and allows the model to better identify what it is looking at. For example, in this paper [55] from 2017 its mentioned that before the context was given of it being a boathouse the model couldn't detect boat and assumed it to be a car but after the context was given of it being a boathouse near a river it was correctly able to recognise the object as a boat. Pyramid Scene Parsing Networks combine the use of local knowledge like in this case along with the global cues to be able to combine them to help it better its ability to identify local objects by using the global cues as context and overall understand what the whole image is showing. The architecture of PSPNET is that, firstly it is given a single input image. This is then passed to an encoder, for example it might be resnet50 as the encoder, this encoder will help extract features and encode the image. It is next passed through a Pyramid Pooling Module, what this does is it's a module for semantic segmentation and acts as global context of the image. The module can then take different sub region representations.

[56] FPN (Feature Pyramid Network) : This is another example of a modern day CNN and this architecture was first created in 2017[56] by Facebook AI Research (FAIR) along side Cornall University and Cornell tech. Each layer has lateral connections similar to UNET, and it is a encoder decoder architecture. This architecture is designed with the intention of being able to recognise different objects which are at different distances / scales.
The architecture is made up of 2 pathways, a bottom up pathway and a top down pathway. The bottom up pathway is what uses the chosen encoder along side convolutional layers and pooling, like in other architectures such as UNET, and will generate the feature pyramid detecting the features in the images at multiple resolutoions allowing the model to capture smaller details whilst still detecting broader objects in the image.
The top down pathway of the architecture is where the model will upscale features from levels of a worse resolution (higher up levels in the pyramid), and then combine together these images with the images of a higher resolution (lower down in the pyramid) through its lateral connections. This merging of different resolution images is what allows FPN to be effective at detecting objects of different scales and or sizes in images.

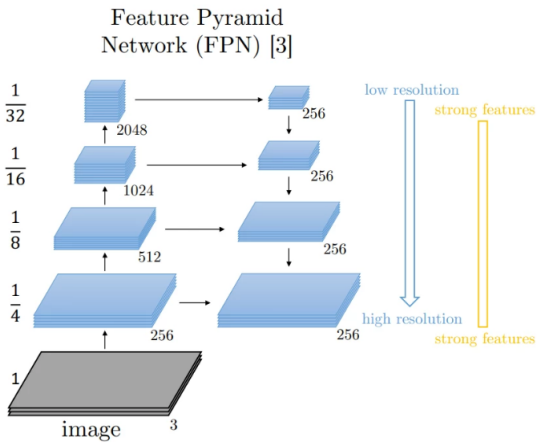


Figure 2.8: FPN Architecture diagram [57]

2.1.5.4 Encoders and decoders

Inside of these previously mentioned models we have different encoders and decoders we can choose from. The encoders role is to interpret the image data effectively to be able to learn certain features within an image, and the decoders job is to decode the encoded image to semantically extract these features onto a higher resolution image.

One very well known CNN that is used as a encoder-decoder within other CNNs such as UNET is one named AlexNet [58]. AlexNet was first introduced in the year 2012 by Alex Krizhevsky. AlexNet was revolutionary at the time as it was the first CNN at that time to use the GPU to boost performance.[59]

AlexNet is made up of 5 convolution layers, 3 max-pooling layers, 2 Normalized layers, 2 fully connected layers and 1 SoftMax layer.
The reason for including normalized layers is due to the fact that after running ReLU activation function it produces an output with no specific range so we normalize the output to all of ReLU outputs are normalized to be within a set range.
A softmax layer [60]in a CNN is most commonly the final layer of a CNN architecture before the output layer and this is because of what it does. The role of the softmax layer is to assign each class in a mutli-cass problem a decimal probability of how likely the model believes the data is to be of each class. A softmax layer is its own neural network layer that is, as stated, last just before the output layer and it must also have the same number of nodes as the output layer

Each convolutional layer is made up of a convolutional kernel filter and also a non-linear activation function, the activation function they use in AlexNet is "ReLU". One potential

limitation of AlexNet that should be of note though is that it has a fixed input size of 227x227 [58] and this is due to the fact that AlexNet contains and makes use of fully connected layers. The reason these fully connected layers result in a fixed input image size is because when a CNN contains fully connected layers every neuron is interconnected and the weights for these neurons are updated as the model is trained, and if we wanted a different, perhaps larger, input image size we would need more neurons with more weights. So the reason this causes the fixed input image size is because we only have so many layers the image is passed through where it is down sampled at each layer to a lower resolution and so the input image has to be below or of a maximum size so that as it is passed through these layers of the CNN it reaches a small enough size to be able to extract all the necessary feature maps from the original input image. In addition to this if the input image is below this 227x227 size AlexNet implements padding to ensure the images are large enough to be passed down through the network.

Another encoder that is very popular is resnet. [61] Resent is a CNN (Convolutional Nueral Network) architecture that is made up of a series of residual blocks (can also be called ResBlocks) with skip connections between them.
When resnet was first created it had solved a big issue at the time which was the vanishing gradient problem.

2.1.5.5 The Vanishing Gradient Problem

The vanishing gradient problem [62] is a problem that occurs during the training of a deep learning model. During training of a neural network after each iteration, or epoch, the weights of each node or neuron are updated based off of the results of our error function, also known as activation function. The issue arises in that as the sequence length increase the gradient magnitude (the gradient being the change in the loss function with respect to the weights calculated via backpropogation [44], this gradient is then used to update the weighs) typically, drastically decreases causing training of the model to significantly slow down.

One large upside of resnet containing skip connections is that convolutional layers can be much faster to train if they contain shorter connections between layers close to the input and close to the output. Another massive benefit of resnet using skip connections is that any layers that degrade the architecture's performance will be skipped by regularization. Regularization is a technique performed in order to prevent over fitting which is when a model becomes over trained on a set of data so much so that it becomes specific to that data and isn't generalized well to learn when it is given new data.

2.2 Existing Food Segmentation Approaches

Semantic segmentation of specifically food is a field that , due to it being quite nieche, has not got a large amount of research in. There are already some existing commerical models as previously mentioned such as MyFitnessPal's 'Meal Scan' feature. These are limited in their use for comparison against the model in this paper because commercial models such as this one are not open source so its not possible to view the medtholodgy that is being implemented and also they do not produce results such as IoU or DICE for a measurement to be able to compare with. There are a some papers on the topic though for example the ones listed below:

- FoodSAM [63]
- SeTR-MLA [64]
- SeTR-Naive [65]

With the best performing being FoodSAM but there is something key to note about FoodSAM and this is that it requires and relies on human input in order to be able to fully segmented and label the foods accurately and even with these producing a MeanIoU of 46.4 (equivalent to 0.464 IoU in the scale 0-1). This is one key limiting factor of the model and is one reason it was not considered to be implemented as the model proposed in this paper should be automated and not rely on human input.

Chapter 3

Methodology

3.1 Data Prep

3.1.1 Selecting the Data set

The first stage of the data preparation stage included searching for , finding and deciding on a data set that would be suitable for the development of the model. Ideally the model would use a dataset that would meet a criteria outlined by having the following attributes: Varied enough to provide a wide variety of foods and has enough occurrences of each food to ensure they all appear enough times throughout the dataset although some foods may not appear as often as others, and a variety of different types of photos, different lighting, angles, distance, sizes.
When researching in order to find a dataset of photos of foods and or meals it was desirable to find a dataset that would meet the criteria which, of course, not all did. For example the Food-11 image dataset. Despite the Food-11 image dataset **food** having 16000+ images having over double the photos of a datset such as FoodSeg103 which will be mentioned shortly, it only had 11 major food categories which, despite the fact that 11 categories of food types is in no way enough, the categories it did have were way to broad. For example one of the categories it mentions containing is 'Dairy Products'. This label is so broad that it would be almost useless in our project as 'Dairy Products could be cheese, which is a clerically dense food, but it could also be semi-skimmed milk, which is not very clerically dense. This isn't even also taking into consideration the obvious problem of one is a solid food whilst one is a liquid making it useless for depth estimation when trying to estimate how much volume / weight there is of the food or item we are looking at which is needed in order to be able to estimate the caloric value of what it is in the image we pass to our model.

After looking into multiple datasets for images of different foods, there were 2 datasets that were considered to be implemented into the model: FoodSeg103, UECFoodPixComplete. The UECFoodPixComplete is a dataset has 10,000 images being 9,000 intended for training

and 1,000 images intended for testing with 103 different food categories. Although UECFoodPixComplete has a larger data set of 10,000 images while FoodSeg has 7114 images, the classes within the UEC dataset are not as well suited towards the typical uk diet and include foods that eaten less often in the uk than appose to other countries such as china, Whereas the FoodSeg103 classes are much more typical of foods eaten in the uk and so are much easier to interpret and see if the model is detecting the correct classes within the image(s). Below are some examples from the FoodSeg103 dataset with their corresponding masks [66]

Figure 3.1: Image Examples of FoodSeg103

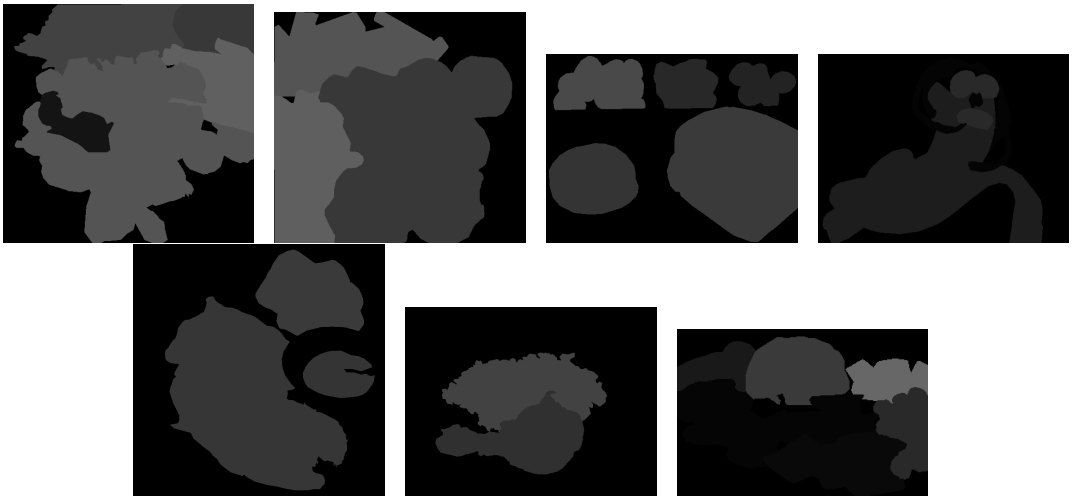


Figure 3.2: Corresponding Annotations for Image Examples of Figure 3.1

Figure 3.3: Figure showing some of the images from the Foodseg103 data set of foods that are more commonly eaten in the UK. (a) Eel [67]

Below some examples of images of foods seen in the both the UCEFoodPixComplete data set and the FoodSeg103 dataset and as seen these foods from the UCE dataset are not as commonly eaten in the uk when compared to the images from the FoodSeg103 dataset and this could make it harder for me to verify the results of my models as I would find it hard to be able to identify those foods and so will most likely struggle to decide whethere the model has correctly or incorrectly segmented the foods in the image.

Another factor that must be taken into consideration is the fact that within the UCE Dataset there would be a very large percentage of the images that contain rice, a disproportionatly large amount in comparison to how often it would appear in phoots of foods / dishes eaten in the UK. This could skew the models training as it may become overfitted specifically for images of foods containing rice.

3.1.1.1 Existing models with FoodSeg103

From the papers with code site where I had found the FoodSeg103 dataset, there were a few models listed below as being models that made use of this dataset giving their Mean IoU. The top of the models with the highest Mean IoU at an IoU of 46.4 was FoodSam [68]. The reason

that this model wasnt chosen to build upon was due to 1 main factor which was the fact that FoodSAM uses and requires human input. Their model requires the user to identify what the foods are on the plate which undermines the point of this model since its supposed to be self sufficient and avoid human input wherever possible except for special circumstances, for example its almost impossible for a model such as the one proposed to be able to detect whether a food is cooked in oil or in butter so requires a human input to clarify. Another smaller factor but worth noting is that with the FoodSAM model in order to improve upon its results would be far outside the scope of this project and would be infeasible.

The FoodSeg103 Dataset has 7117 images across 103 categories of foods with detailed masks/ground truths for each image. And the first task was to pre process the data so it would be suitable for our model.

3.1.2 PreProcessing

Firstly all the images from the dataset were be default in a JPEG format, the ground truths were already in PNG format and in addition to this, unlike JPEG, PNG format images are losses meaning that they dont lose detail when they are compressed and so can be fully recontrsuted from their compressed format. When compressing a JPEG image on the other hand it is losses meaning some of the data in the image will be lost and so this may cause some key details to be lost and thus effecting the efficiency of our model due ot poor data input.

So due to this the first task I completed was to converted all the images from JPEG to PNG using matlab. Next I used unique() command in matlab to see how often each class appears. It was important to do this as it wouldn't be good to implement a data set where only 2 classes appear a disproportionate amount of times compared to the other classes. It's unlikely to have all the classes be of an even number of appearances but having them be spread out over a somewhat normal distribution allows for us to have better training data for the model. An example of a poor set of training data would be one where carrots only appeared in 50 images out of for example 5000 images, of course the model will then unfairly struggle to segment carrots as appose to chicken if that appeared in 750 of the 5000 images in the training data set. The count for how often each class appears provided a relatively normal distribution with common foods such as chicken appearing X number of times.

3.1.2.1 Splitting the Data set

Once the set of images had been compiled into all PNG format. I then took the masks of the images and the images themselves and split the data into 3 categories: Training Data, Testing Data, and Validation Data, split 60:20:20 respectively. This was done as I want to train the data with the largest set possible to give it a wide enough range of data to use to learn from, then leaving us 1/5 of the data to be used to test and validate the results of our model after being trained with the training data set.

3.1.3 Building our model

The model was started with a UNET decoder encoder architecture and the model was first ran with 40 epochs on the training section of the FoodSeg103 dataset using the resnet encoder to test that the model works as expected before running to test different encoders an decoders against each other. In our model it begins firstly by setting the file locations for the 3 data categories mentioned above, training, testing, and validation. Once this is done the next step is to then creat a dataset class which contains all the names of the classes so the model knows which numbers belong to which labels for example if 1 = candy.

Next the model takes all the data in our training set and performs augmentations to the images in our data set. Augmentations are performed on the model because we need as many inputs (or images) as we do parameters for example, if we have 60 million parameters its quite difficult, borderline impossible, to have 60 million unique images, augmentations is how the model compensates for this. The data set's images are manipulated in a NON linear manner to create , what seems to the model to be, new data. These augmentations to the images could include non linear intensity shifts, aspect ratio adjustments, increasing or decreasing the contrast of the data, and more. The augmentations must be non linear so the model doesn't treat the new adjusted images as the same as the original image.

This is when its time to actually move into the training of the model. For the model it is using CNNs to semantically segment the images to detect each class of food it is viewing in the image and to do this the model is made up of a CNN architecture which within that will implement an encoder/decoder. These are imperative to get right in order to produce a model that is well suited towards the problem and can be effective whilst still being efficient.

3.1.4 Choosing the encoder-decoder

It's crucial to the methodology to have a encoder and decoder combination that is well suited towards the problem and will be most effective at completing the tasks with as little overhead as possible. For example I may consider popular encoders such as resnet and dense net. It is important to note here that whichever methodology is chosen for the encoder will be the same architecture that is used to decode the feature maps. This is due to the fact they are their own unique algorithms so if you were to encode your data with 1 encoder you need the same algorithms to be able to decode that data set as only that algorithm can understand its own encoding

Both of these factors lead me to my first experiment which was to compare different

encoder decoder algorithms to find the most effective one for our solution and then once this was completed, it would next be ready to run the CNN architecture with these different encode-decode algorithms to find which would lead to the fastest and most efficient results. During our experiment we first would run each architecture encoder/decoder setup with just 1 of our classes, this was in order to keep the training as simple as possible and attempt to remove any outside factors that could sway the performance of the different encoder-decoders. For example the algorithm "X" is particularly well suited towards smaller much more finer details and so if we were to test on all of our classes this encoder would perform much better on food such as pees where they are much smaller but it also may perform worse on broccoli as it could very well interpret each head on the broccoli as a very small food which is of course incorrect. We also want to makes sure that we are training with a class that has plenty of appearances to give the encoder-decoders the best change possible of showing their capabilities.

The class we chose to test on to begin with was bread as it was the most often appearing class in the data set, appearing a total of X times across our entire dataset providing a good base to test on. When testing on a single class we don't want a class that doesn't appear very often as this would skew our results and may not be applicable when applied to more common classes in the dataset.
Firstly the modelw was ran with resnet encoder-decoder on the FPN architecture just to ensure the code ran and worked as expected in which it did. So once this had completed as expected without crashing it was then ran all of our encoder-decoder algorithms on the UNET backbone. Keeping the same backbone when carrying out our experiments allows us to fairly compare each encoder-decoder algorithms. The encoders and decoders that were tested are listed below:

- Densenet201
- Resnet50
- Resenet152
- VGG19
- VGG16

From the results of our experimentation using different encoders and decoders it was found that resnet50 had the best overall performance across 200 epochs providing the most consistent results and so resnet50 will be implemented as the encoder moving forward in the model from this stage.

3.1.5 Choosing the architecture.

Once these different encoder-decoder algorithms with the UNET CNN architecture had completed I then next moved onto to beginning experimentation to determine what architecture would be appropriate for the problem at hand and whether the model should implement UNET, UNET++, FPN (Feature Pyramid Network), or another architecture. I need one that is well suited towards the problem, being efficient whilst having a low overhead. The architecture that the model would run on is very important as if the architecture for the model was not efficient for the problem it would limit the models performance in the long run despite how efficient other areas of the model may have become. So now using the encoder-decoder that we had chosen in our last experiment to be resnet50 I then ran it across following different architectures:

- FPN
- UNET
- UNET++

From the testing with different models it was found that UNET++ provided the best overall statistics and so moving forward the model will be a UNET++ based architecture with resnet50 as the encoder / decoder.

3.1.6 Testing and Improving the model

Once the Architecture and encoder-decoder that was the most effective for the problem had been determined the next stage was to attempt to begin training and testing this same model but with ALL the classes from the dataset. So far the model had only been tested and train on a subset of the dataset , specifically on only the bread class as this was the most common class in the dataset.. Moving to training and testing the model on the dataset as a whole would give a much more realistic view of this capabilities as here lots of different possible scenarios are being accounted for such as classes which appear very often , some which appear only a small percent of the time, some which are broader categories including different specific foods under one same name, different lighting. Testing in this way across the entire dataset allows me to test the versatility of the model.
When trying to do this the model ran into a computational limitation. Upon attempting to run the model with all 103 classes it was unable to complete and repeatedly crashed. This was found to be due to the amount of memory required for this and the model was limited by computational power of the setup the model was being ran on. To account for this the model instead was trained on the 31 most common classes to train and test this model. The model was tested and trained using the top 31 food classes, this was 32 overall classes as the

background is one of the classes. The reason for the number 32 is it is conventional to use powers of 2 and as the number of classes was taken down from 103 to 64 which was still too much for the amount of memory available to handle so it was lowered further to 32 and this was manageable for the system.

When running the model with these 31 most common classes it was found that when being ran on the UNET++ with the resnet50 encoder the model was only able to produce results for segmenting the following 10 classes of food.

- bread
- broccoli
- carrot
- chicken-duck
- cilantro mint
- corn
- ice-cream
- steak
- strawberry
- tomato

It was theorised that this could possibly caused by 1 of 2 reasons. The first possible cause being due to the fact the model was using neuman boundary conditions when performing padding to the input data set. Our input data set for training contained images that had been compressed down into a 256x256 resolution, upon training with this data set with the compressed version of the images the model requires larger images, specifically of a minimum size of 480x384.
The reason the model had notionally been tested with an input image size of 256x256 was because when compressing the images down to a smaller resolution it would in turn allow the model to run faster due to the fact that at each layer the model has a smaller images size to cover with the kernel(s) and thus allowing the model to complete each layer in the CNN much faster and so complete the overall training much quicker.
The reason that the image was padded within the model is because as the model progresses down the contracting side of the UNET++ architecture the image becomes down sampled to a lower resolution at each layer and so if the input images from the data set when training

are not of a large enough resolution the model wont be able to continue down sampling the image at every layer as eventually the image will become to small. To tackle this the model implements padding. This is where the image is padded with pixels in order to meet the minimum input image resolution of 480x384. There are few different ways it can choose how it is going to pad the images and neuman boundary conditions is one of them.

[69] The Neuman boundary condition is where pixels from the edges of the image that need to be padded are taken from the image itself and reflected. For example if the image required extra padding of 64 pixels on each side, using Neuman boundary conditions, it would take 64 pixels inward from that same edge of the image and mirror these to then pad that side with these pixels in order to make up the missing space. This can cause some errors especially if the image isn't padded uniformly on the top and the bottom of the image. In the case of the model that was ran using 256x256 images, in order to reach the minimum resolution of 480x384 the model padded the images horiztonally by 64 vertically at each edge and then by 112 pixels on each horizontal edge and due to the fact that the image is being padded in a non-linear fashion meaning that the content of the image on the horizontal edges now represents a larger proportion of the image as compared to before this can effect the model negatively. One potential way this can be problematic for testing is if in come classes the foods of interest are more often than other classes positioned closer to the edges.
This could potentially skew the IoU score and the Accuracy scores as now the number of pixels in the image that are that of the food of interest my have increased in a non-linear fashion which would impact the accuracy and IoU as that isn't actually how the food of interest looked in the original image.

This could negatively effect the models confidence score and this be a potential cause for why the model is only segmenting certain classes from the dataset as appose to all of them. This would be because if certain food classes are more commonly on the edge of the image then they would experience more of this mirroring effect which would in turn, potentially cause there to be more pixels in the image of that food class making it easier for the model to be trained on these classes and struggle to be trained on the remaining 21 food classes.
Another potential cause for the lack of classification for the other 21 images could be due to the size of the dataset for the other images, for example, the bread class had by far the most images in the data set at a total of 1405, whereas the 10th most common class Cilantro Mint had a total of 900 images and the 28th most common class in the dataset noodles had a total of 262 images in the dataset. This variance in the number of appearances of each dataset in the model could cause the model to have poorer generalisability with some classes causing it to, for the 21 classes that it did not segment, have a confidence score that fell below the threshold required in the code for the model to deem pixels to be of a certain class of food.

In order to overcome these errors caused by an image size below the minimum the model was

then re-ran with an input image size of 512x512 in order to remove the issue of the images requiring padding and thus effecting the testing output. Upon doing this we found that the model still only segmented 10 classes out of the top 31 most common classes from the dataset. Since the model had been ran again but this time with the images being increased to a resolution of 512x512 and thus the model is no longer padding the image it can safely be said that the reason for the only 10 classes being segmented is not due to the Neuman boundary conditions.
Upon reviewing the output for the model with images of the new 512x512 resolution we saw that the 10 classes that were segmented were almost identical bar 1. This time instead of the same 10 classes it classified 9 the same but sauce instead of corn. As previously stated there are multiple reasons this could potentially occur, segmenting the same 10 classes. As for why it segmented corn instead of sauce would suggest its related now to the images not containing padding and so in the non padded version of the images it may be that the sauce is more prominent than the corn.

Despite not being able to within the scope of this project overcome this occurrence and investigate further the cause of this issue the model was ran by only being trained on those 10 classes it produced from the 512x512 images as this may allow the model to achieve higher IoU and accuracy and show that if this issue was overcame and if there was less interference in the training of the model then it would allow the model to be better suited towards those classes. The model was ran on 2 cases, the most common 10 classes which had the largest data set for those 10 classes and it was also ran on the 2nd most common 10 classes those being classes the most common 10 after those first 10.

When viewing the output for the 10 most common classes from the dataset it was clear that the top 3 most common classes, bread, carrot, and chicken-duck were far above the other 7 in terms of IoU. This would be most likely due tot he fact that these classes have a far more images for them than the other classes in the dataset cause the model to be heavily skewed toward these datasets and not being able to allow for enough variance within its training for the other classes in the dataset. This may also cause the confidence score to be lower for the other 7 classes which would in turn explain the cause of the poor segmentation of these 7 classes as the model isn't confident enough to classify said pixels that DO belong to that class as part of that class.

Upon the second run of the mode but this time using the next 10 most common classes from the dataset it was clear that there were still errors in the model. Despite being trained on the 11th-20th most common classes the model was still only producing segmentation results for the top 10 most common classes. It wasnt clear why this was happening during testing and was a key limitation of the model that would had been infeasible to be able to continue to investigate in order to overcome and solve.

As seen in results from these 3 different tests, 32 classes, the 10 most common, the next

10 most common, each show different possibilities for the model namely that when the model is narrowed down it focuses on less classes, typically the top 10 most common classes and or 1/3 of the overall number of classes that it is trained on but the IoU and accuracy for this model is higher than with the 32 classes showing the potential the model has if the challenge of it focusing only on 1/3 of the dataset was to be overcome. One potential reason that the model would perform better with the 32 classes as appose to when being trained on just 10 classes is that when being trained on the 32 classes it has more adversarial classes to train on. By this I mean with the 32 classes the model is able to train itself on more classes meaning when it comes to interpret a certain class because it has been trained on more classes it can better estimated what that class is not and therefore narrow down the possibilities for the classification.

3.1.7 Analyzing the model's performance

Once the model has completed a training cycle and then testing, its best next to extract some stats from the model which can be exported as a comma-separated values (CSV) file. Here I'm going to be discussing these stats, what they mean and why they are important for the analysis of the model.
Firstly we have FN which is False Negatives. This is the number of times the model has labelled a pixel as NOT being part of a certain class but in the ground truth it was part of that class.
Next we have FP standing for False Positives. This is similar to FN but instead is the number of pixels that the model classified as being part of a certain class yet in the ground truth they were NOT part of that class.
TN which stands for True Negatives is the number of pixels where the model said the pixel was NOT part of a certain class and in the ground truth it also was not part of that class in other words it correctly segmented that pixel to not be a class it wasn't.
TP is alternatively True Positives and as expected this is the number of pixels where there model segmented a pixel as being part of a class and in the groud truth that pixel WAS part of that same class, in other words its when the model has correctly segmented a pixel of the image to be of a class that it is part of. Then there is sensitivity which is defined as

$$\frac{TP}{TP + FN} \tag{3.1}$$

This is, simply put, is as a fraction what proportion of the pixels in the image that DO belong to a certain class did the model correctly detect as being part of that class within in the image. This is useful as its effectively a score showing us of the however many pixels that are bread in the ground truth for example, what percent of the pixels we said were bread were actually bread in the groundtruth.

Next we have specificity which is defined as:

$$\frac{TN}{TN + FP} \tag{3.2}$$

This being he True negatives divided by the sum of Total negatives and False positives. Here in simple terms this gives, as a fraction, of the pixels that the model predicted to NOT be of a certain class, what % of these pixels were actually NOT part of that class in the groundtruth. The next important statistic that shall be used for analysis of the models performance for each class will be Precision and this is defined as:

$$\frac{TP}{TP + FP} \tag{3.3}$$

Precision simply means of the area that the model determined to be part of a class what % of that was actually part of that class in the ground truth. All of these statistics are useful to be able to conclude an overall idea of how well or poorly the model is performing for example, if the model has a very good precision score meaning it rarely produces false positives and only segments pixels that are indeed of that class but then has a very poor Specificity score then this may indicate that the model needs to be tested using a higher confidence value as pixels that are in fact part of that class are not meeting the confidence score and so not being segmented, for example.

3.1.8 Calculating Volume and Calories

At this stage we have our model able to semantically segment different foods apart from each other on a plate of food from a single point of view photo but now we need to estimate the caloric value of the food on the plate. The first and most important step in doing this is we need to be able to estimate how much food we are viewing on the plate specifically the weight of that food. Now because the model is attempting to estimate this weight the it requires 2 measurements or metrics for the food(s). The estimated volume and also the estimated density since the weight will be estimated via:

$$EstimatedWeight = EstimatedDensity \times EstimatedVolume \tag{3.4}$$

Firstly the model is going to first calculate the estimated volume of the food(s) in the image. Due to the time frame within which this project is required to be completed it was not feasible to code from scratch a unique original depth perception model for food and train it. Also its noteworthy to mention that to obtain the most accurate and reliable estimated volume of the food it would be optimal to use a 3D reconstruction of the food. This alone is very difficult and on top of this especially performing this from just a single image is extremely difficulty and far out of the scope of this project. As a solution to this problem I decided to implement a pre trained model, and one that is specifically trained on foods to be

better suited towards our problem rather than a general depth perception model. The pre-trained depth perception model that was implemented was FVEsitmator, more specifically I'm going to be implementing an already existing version of it by Alex Graikos [70]. This is a pre-trained depth perception model with predefined weights that is intended for the depth perception of different food items. An image of the volume estimation model's architecture can be seen below:

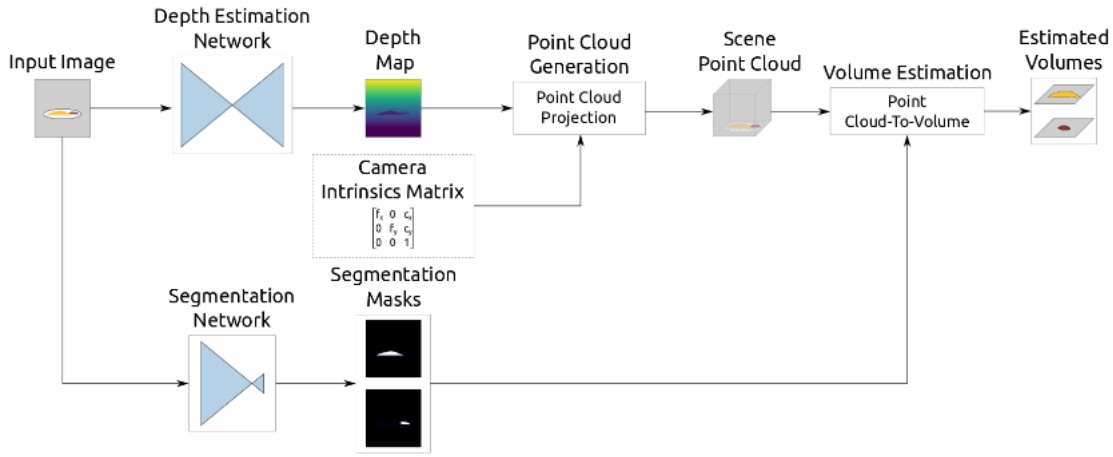


Figure 3.4: Architecture of the FVEstimator [70]

The models general approach is to firstly run the image through its depth estimation network in order to produce a depth map for the image, which is then used to create a point cloud which is then used, in turn with the segmentation mask that is produced from the image also being put through a segmentation network, to produce a volume estimation for the segmented food(s). The way the model was integrated with the FoodVolumeEstimator in order to estimate a caloric value was by taking the outputed food Segmentations and the estimated volumes for each of the segmentation's for each image and overlay the segmentation with the ground truth masks for that same image in order to be able to label each region in the depth map from the FVE as the food(s) they are in order to be able to estimate the caloric value of the food(s) in the image. Alternatively the segmentation's from the custom model that was made could be used in place of the segmentations in an ideal model but in reality the ground truth masks have been used as they are known to be perfectly accurate and can allow for better fairer testing of the FVE as the caloric number produced will solely rely on the FVE instead of potentially error due to the FVE or the segmentation model's output where it would not be possible to easily distinguish which has caused what degree of error. Now having the estimate volume for each food and the values being labelled as their food class this is then used in conjunction with the AverageFoodDensity table refer to here to calculate the estimated weight of the food via the equation which was given above:

$$EstimatedWeight = EstimatedDensity \times EstimatedVolume \tag{3.5}$$

The reason it is required to have an estimated weight is because foods are standardised in terms of weight meaning calories are given in relation to weight, typically per 100g. Once there is an estimated weight its required to find this foods estimated per100g calories and perform the following equation:

$$EstimatedCalories = \frac{EstimatedWeight}{100} \times Caloriesper100g \tag{3.6}$$

This finally produces an estimated caloric value for the foods as well as for those calories extracting estimated values for the food(s) Carbohydrates, fats, and proteins. The way this was implemented into the model was via a MatLab script. There were 2 folders one with the segmentations from the FVE and one with the ground truths from the FoodSeg103 Dataset. Each image from the dataset has multiple segmentations one for each food in the image from the FVE. Each FVE segmentation image is taken and turned into a binary image and then overlayed with the groudntruth masks to find what class ID is labelled in the regions marked as 1. this is because in the segmentation images produced by the FVE the entire image is black except for where the food is so representing 0 so if we take the regions where the value is greater than 0 we are only looking at the region(s) that contains segmented foods from the FVE. Then from this the total volume for each classa is totaled and there is then a total volume for each label of food presented in the image. This is then used in the above equations with the fooddensitytable and a calorie value table in ordert to esitmate the final caloric value of the food(s) in the image.

Chapter 4

Results

4.0.1 Implementations

The implementations for this project were as follows: The segmentation model to segment and label the foods in the images was coded in python 3.10 using the following libraries:

- torchvision version 0.5.0
- pretrainedmodels version 0.7.4
- efficientnet-pytorch version 0.7.1
- timm version 0.9.7

The separation of the dataset to begin with was done in Matlab version ....

The Foodvolume estimator model was ran on python 3.6 and inlcuded the following python libraries:

- numpy==1.16.3
- pandas==0.24.2
- opencv-python==4.1.0.25
- scipy==1.2.1
- tensorflow==1.13.1
- keras==2.2.4
- scikit-learn==0.21.1
- scikit-image==0.16.2

- matplotlib==3.0.3
- pyntcloud==0.1.2
- pythreejs==2.1.1
- IPython
- Flask==1.1.1
- fuzzywuzzy==0.18.0

All code and experiementation was ran on a computer with the following specificaitons:

- DELL 5000 PC
- ZEON W2245 CPU
- NVIDIA RTX 60000 ada gene 18000 cores

DELL 5000 with ZEON W2245 CPU

4.0.2 Comparing Encoder Decoder combinations

The first major task in building this model was to be able to correctly identify which encoder or decoder would be best suited towards solving the problem. The first set of experiments that was ran was to run the model with different encoder-decoder to compare which would perform best within the dataset. The experiment was performed with each encode-decoder being ran on the same architect (UNET) in order to ensure consistent results across the different encoder-decoders. The ecoders and decoders that were teste on the UNET architecture were:

- VGG16
- VGG19
- RESNET50
- RESNET152
- DENSNET201

Each encoder-decoder would run on UNET and would be trained on the entire training and testing datasets but would only segment and learn from the bread class. This was done to firstly ensure consistency as all encoder-decoders are training on the same class within the dataset and also because bread was the most common occurring class within the dataset and so gave each model the largest number of images to learn from.

Model	Best Train IoU	Best Val IoU	Best Test IoU	Best Val Epoch	TrainIoU
densenet201	0.9041	0.6379	0.6407	84	0.8586
resnet50	0.8800	0.6421	0.6311	75	0.7996
resnet152	0.8726	0.6337	0.6209	109	0.8053
vgg16	0.8635	0.6424	0.6297	127	0.8310
vgg19	0.8547	0.6367	0.6452	189	0.8482

Model	Train Dice AUC	Val Dice AUC	Train IoU AUC	Val IoU AUC
densenet201	0.1025	0.2906	0.8304	0.6041
resnet50	0.1247	0.2876	0.7958	0.6085
resnet152	0.1397	0.3069	0.7721	0.5855
vgg16	0.1370	0.2938	0.7770	0.6016
vgg19	0.1393	0.3013	0.7715	0.5918

From this testing, as seen in the results tables above, it was found that with the UNET architecture VGG16 had the highest Validating IoU in a single epoch of 0.6424 during training but upon being tested on our testing data I found that it was VGG19 with the highest testing IoU of 0.6452. Although these had the best single epoch and best testing values its important consider their average performance or area under curve, meaning if we were to plot a graph with the x axis being the epoch number and the y axis being the IoU which has the largest area under the curve and therefore the best IoU overall. This is important to consider because in an eviroment such as this model with the given dataset where its somewhat noisy and the values can vary from epoch to epoch taking the highest single epoch may be such an outliter that overall that encoder didnt perform very well it just had one outlier that perforemd well. Upon considering the Area under curve IoU for each endoer it was found that it was resnet50 that had the best average IoU score ,IoU of validation data, at a value of 0.6085 Therefore when taking into consideration all epochs it was assumed that resnet50 would be the most effective encoder/decoder for segmenting the data in this model.

4.0.3 Comparing Model Architectures

Once the correct encoder-decoder had be selected for the model moving forward the next decision to be made was which architecture would this encoder-decoder be ran on. In order to determine this the next experiment that was to be ran was to test different architectures with resnet50. resnet50 was then trained and test with the following architectures:

- FPN
- UNET
- UNET++

The results are show below

Architecture	Avg Val IoU	Highest Val IoU	Test IoU
FPN	0.596625	0.632799	0.631506
UNET	0.608513	0.639384	0.631114
UNET++	0.618119	0.644158	0.625325

From the testing with these different architectures it was found that the model with the highest average Validation IoU score was UNET++ at 0.618119, UNET++ also had the highest single epoch IoU Validation score of 0.644158, despite this during our testing data UNET++ had the lowest IoU score of 0.625325 with FPN having the highest at 0.631506. From this it is going to be most effective to go with UNET++ as it has the highest single epoch IoU score along side the best Average IoU / Under the curve IoU score showing it to have the most potential.

4.0.4 Testing and Improving the Model

The next experiment that was performed once it was decided that the model would use UNET++ and Resnet50 moving forward was to run the model on the the entire dataset including all 103 different classes from the FoodSeg103 dataset. This would be to botain an initial benchmark of how the initial model would perform with the FoodSeg103 dataset. Upon attempting this experiment the model ran into a technical limitation of how much memory was required to run such an experiment. Because of this the model was then attempted to be ran with a lower number of class that being the most common 64 classes, 64 because its a common to use numbers that are base 2, and still this exceed the machine's memory and so it was lowered again to 32 classes which the machine was able to process without crashing or erroring. Upon the completion of testing and training with the most common 32 classes a table of results was produced which can be seen below:

	FN	FP	TN	TP	Sens	Spec	Prec	IoU	Dice	Acc
asparagus	438	0	65098	0	0.000	1.000	n/a	0.000	0.000	0.993
biscuit	576	0	64960	0	0.000	1.000	n/a	0.000	0.000	0.991
bread	1025	1815	60677	2019	0.627	0.970	0.367	0.251	0.292	0.957
broccoli	129	352	63958	1097	0.846	0.994	0.505	0.441	0.486	0.993
cake	1174	0	64362	0	0.000	1.000	n/a	0.000	0.000	0.982
carrot	200	543	63688	1104	0.821	0.991	0.402	0.339	0.379	0.989
cheese-butter	356	0	65180	0	0.000	1.000	n/a	0.000	0.000	0.995
chicken-duck	885	2559	60226	1866	0.598	0.960	0.282	0.198	0.224	0.947
cilantro-mint	276	570	64255	435	0.567	0.991	0.239	0.150	0.183	0.987
corn	52	150	64850	484	0.892	0.998	0.513	0.464	0.497	0.997
cucumber	525	0	65011	0	0.000	1.000	n/a	0.000	0.000	0.992
egg	449	0	65087	0	0.000	1.000	n/a	0.000	0.000	0.993
fish	746	0	64790	0	0.000	1.000	n/a	0.000	0.000	0.989
french-beans	416	0	65120	0	0.000	1.000	n/a	0.000	0.000	0.994
french-fries	426	0	65110	0	0.000	1.000	n/a	0.000	0.000	0.994
ice-cream	397	996	63539	604	0.560	0.984	0.212	0.148	0.175	0.979
lemon	479	0	65057	0	0.000	1.000	n/a	0.000	0.000	0.993
lettuce	669	0	64867	0	0.000	1.000	n/a	0.000	0.000	0.990
noodles	498	0	65038	0	0.000	1.000	n/a	0.000	0.000	0.992
onion	402	0	65134	0	0.000	1.000	n/a	0.000	0.000	0.994
orange	214	0	65322	0	0.000	1.000	n/a	0.000	0.000	0.997
pepper	398	0	65138	0	0.000	1.000	n/a	0.000	0.000	0.994
pie	1086	0	64450	0	0.000	1.000	n/a	0.000	0.000	0.983
pork	1416	0	64120	0	0.000	1.000	n/a	0.000	0.000	0.978
potato	1529	0	64007	0	0.000	1.000	n/a	0.000	0.000	0.977
rice	1030	0	64506	0	0.000	1.000	n/a	0.000	0.000	0.984
sauce	1148	0	64388	0	0.000	1.000	n/a	0.000	0.000	0.982
sausage	290	0	65246	0	0.000	1.000	n/a	0.000	0.000	0.996
steak	773	1564	61913	1286	0.580	0.976	0.343	0.235	0.266	0.964
strawberry	105	394	64584	453	0.804	0.994	0.299	0.247	0.282	0.992
tomato	201	656	63980	698	0.763	0.990	0.275	0.218	0.252	0.987

The table produced includes all 32 classes down the left hand side of the table labelling each row. There is 31 classes of food as one of the classes was background making up the 32 classes total.

Taking a detailed look at the resulting results table for the 256x256 testing its clear that only 10 classes of the 31 have been segmented by the model, as mentioned during the methodology section this could be caused by a few reasons some of which being: potentially

there isn't enough training data for the model to reach a confidence score that is of above the confidence value thresh-hold in the model, it may also be due to the fact that some classes are being miss classified as other classes , e.g. cheese as bread so then from the model's perspective it has even less data to train on the misclassified class.
Looking at the class with the largest data set bread , it has a precision of only 0.367 and a sensitivity of 62.7%. This would imply only about 36.7% of the pixels the model predicts to be bread are actually bread and this may be due to the miss-classification of other classes for example biscuit being mistaken as a slice of bread.

Another interesting class' results to take a look at is the results produced for the broccoli class which has a sensitivity of 84.6% meaning that of the pixels that were labelled as broccoli in the ground truth mask the model correctly predicted 84.6% of those as broccoli.
This could be a cause of multiple causes, it could perhaps be due to the images of broccoli being clearer compared to for example the bread class, it may also be a cause of the fact that naturally broccoli is, by the nature of it, quite a unique class due to its shape, texture and colour. This may help the model better segmented the broccoli correctly as broccoli as it doesn't resemble many other foods if any at all in the data set.

There was a major error in these results though, as the images inputted into the mode were of a size of 256x256 this caused a massive issue because the model required a minimum input image size of 480x384. So because the input images were below this they were padded with Neuman boundary conditions which is when for example horizontally the input images were 112 pixels short at the top and bottom 224 total. So if you take the top side of the image it will move inwards from the top border by 112 pixels take that section of the image and mirror it on the top and so on on all sides of the image.
This causes issues because it will manipulate what / how much of a food is show in the image as if i an image is near the edge then it will be mirrored or a portion of it will be and then by the time it has mirrored on all edges of the image it will cause the amount of the food(s) in the image to be effected and so will effect the estimated volume, weight, and of course calories.

In order to overcome this the images were no longer down sampled and were inputted at an image size of 512x512 in order to overcome this. The results from this model can be seen here below:

	TP	TN	FP	FN	Sens	Spec	Prec	IoU	Dice	Acc
bread	8699	241105	8864	3476	0.711	0.964	0.263	0.197	0.227	0.953
carrot	4603	255384	1542	615	0.859	0.994	0.397	0.354	0.386	0.992
chicken-duck	7597	240452	10689	3407	0.627	0.958	0.246	0.176	0.199	0.946
sauce	0	257553	0	4591	0.000	1.000	n/a	0.000	0.000	0.982
tomato	2916	256420	2128	680	0.801	0.992	0.300	0.249	0.280	0.989
potato	0	256026	0	6118	0.000	1.000	n/a	0.000	0.000	0.977
steak	5088	247464	6446	3146	0.588	0.975	0.238	0.152	0.176	0.963
broccoli	4358	256503	734	549	0.857	0.997	0.487	0.429	0.465	0.995
ice-cream	2050	256524	1615	1955	0.456	0.994	0.300	0.180	0.205	0.986
cilantro-mint	1667	257590	1708	1178	0.559	0.993	0.260	0.162	0.195	0.989
rice	0	258024	0	4120	0.000	1.000	n/a	0.000	0.000	0.984
pork	0	256482	0	5662	0.000	1.000	0.000	0.000	0.000	0.978
lemon	0	260229	0	1915	0.000	1.000	n/a	0.000	0.000	0.993
lettuce	0	259468	0	2676	0.000	1.000	n/a	0.000	0.000	0.990
strawberry	1951	259008	903	281	0.860	0.996	0.336	0.295	0.324	0.995
pie	0	257798	0	4346	0.000	1.000	n/a	0.000	0.000	0.983
cucumber	0	260044	0	2100	0.000	1.000	n/a	0.000	0.000	0.992
onion	0	260536	0	1608	0.000	1.000	n/a	0.000	0.000	0.994
corn	0	259999	0	2145	0.000	1.000	n/a	0.000	0.000	0.992
cake	0	257448	0	4696	0.000	1.000	n/a	0.000	0.000	0.982
pepper	0	260552	0	1592	0.000	1.000	n/a	0.000	0.000	0.994
cheese-butter	0	260719	0	1425	0.000	1.000	n/a	0.000	0.000	0.995
french-beans	0	260478	0	1666	0.000	1.000	n/a	0.000	0.000	0.994
fish	0	259159	0	2985	0.000	1.000	n/a	0.000	0.000	0.989
biscuit	0	259838	0	2306	0.000	1.000	n/a	0.000	0.000	0.991
egg	0	260348	0	1796	0.000	1.000	n/a	0.000	0.000	0.993
asparagus	0	260392	0	1752	0.000	1.000	n/a	0.000	0.000	0.993
noodles	0	260154	0	1990	0.000	1.000	n/a	0.000	0.000	0.992
orange	0	261289	0	855	0.000	1.000	n/a	0.000	0.000	0.997
sausage	0	260985	0	1159	0.000	1.000	n/a	0.000	0.000	0.996
french-fries	0	260441	0	1703	0.000	1.000	n/a	0.000	0.000	0.994

Upon completing running the model with an input image size of 512x512 we could now see without padding how the model treated the data and what the results were. As seen in the tables shown above now without the padding being added due the inpit images now being of a large enough size it can clearly be seen that the IoU has increase in in some classes such as carrot increasing from 0.339 to 0.354 and in tomatoes going from 0.218 to 249 while in some classes the IoU decrease such as in broccoli going from 0.441 to 0.429 but in some of the classes where IoU has fallen accuracy has increase such as in broccoli despite the drop in

IoU the accurate increase from 0.993 to 0.995. These changes as previously mentioned were due to the fact that now the images were not receiving padding and so these results are now true to the actual original images. This could also partially be due ot the fact that now the model is recieveing more detailed input images as they are at a higher resolution, twice as high.

There was an a very promising result from the 32 class experiment when modifying the results slightly as seen below:

	FN	FP	TN	TP	Sens	Spec	Prec	IoU	Dice	Acc
asparagus	438	0	65098	0	0.000	1.000	n/a	0.000	0.000	0.846
biscuit	576	0	64960	0	0.000	1.000	n/a	0.000	0.000	0.790
bread	1025	1815	60677	2019	0.627	0.952	0.826	0.532	0.618	0.892
broccoli	129	352	63958	1097	0.846	0.979	0.869	0.744	0.819	0.971
cake	1174	0	64362	0	0.000	1.000	n/a	0.000	0.000	0.733
carrot	200	543	63688	1104	0.821	0.982	0.848	0.708	0.793	0.968
cheese-butter	356	0	65180	0	0.000	1.000	n/a	0.000	0.000	0.897
chicken-duck	885	2559	60226	1866	0.598	0.969	0.815	0.537	0.606	0.902
cilantro-mint	276	570	64255	435	0.567	0.989	0.821	0.489	0.600	0.960
corn	52	150	64850	484	0.892	0.986	0.908	0.813	0.872	0.975
cucumber	525	0	65011	0	0.000	1.000	n/a	0.000	0.000	0.891
egg	449	0	65087	0	0.000	1.000	n/a	0.000	0.000	0.852
fish	746	0	64790	0	0.000	1.000	n/a	0.000	0.000	0.761
french-beans	416	0	65120	0	0.000	1.000	n/a	0.000	0.000	0.891
french-fries	426	0	65110	0	0.000	1.000	n/a	0.000	0.000	0.800
ice-cream	397	996	63539	604	0.560	0.969	0.715	0.434	0.514	0.926
lemon	479	0	65057	0	0.000	1.000	n/a	0.000	0.000	0.911
lettuce	669	0	64867	0	0.000	1.000	n/a	0.000	0.000	0.883
noodles	498	0	65038	0	0.000	1.000	n/a	0.000	0.000	0.798
onion	402	0	65134	0	0.000	1.000	n/a	0.000	0.000	0.930
orange	214	0	65322	0	0.000	1.000	n/a	0.000	0.000	0.900
pepper	398	0	65138	0	0.000	1.000	n/a	0.000	0.000	0.910
pie	1086	0	64450	0	0.000	1.000	n/a	0.000	0.000	0.777
pork	1416	0	64120	0	0.000	1.000	n/a	0.000	0.000	0.773
potato	1529	0	64007	0	0.000	1.000	n/a	0.000	0.000	0.859
rice	1030	0	64506	0	0.000	1.000	n/a	0.000	0.000	0.837
sauce	1148	0	64388	0	0.000	1.000	n/a	0.000	0.000	0.903
sausage	290	0	65246	0	0.000	1.000	n/a	0.000	0.000	0.867
steak	773	1564	61913	1286	0.580	0.984	0.872	0.529	0.598	0.911
strawberry	105	394	64584	453	0.804	0.978	0.850	0.692	0.790	0.958
tomato	201	656	63980	698	0.763	0.984	0.823	0.643	0.743	0.967

In the table shown above what was changed was after the final output from the model for each output segmentation the labelling were changed iterating over the whole list of the 32 labels until a label that produced the best result was found and that was set as the label. Upon doing this the IoU and Dice scores are improved immensely. For example the IoU for broccoli increasing from 0.429 to 0.744.

It was clear in both sets of results that there was a rather impactful flaw in the model in the fact that the model seemed to only focus and train itself on 1/3 of the classes it was intended to be trained on it was theorised that a potential cause of this was the Neuman boundary conditions but after having the padding via Neuman boundary conditions removed by rerunning the model with a large enough input image size this was proven to not be the causing factor.

Another possible cause that was considered was the fact that these classes have both a larger data set size when compared to some smaller classes further down the list but also the fact that potentially the data for these classes was better suited towards the model meaning that they perhaps had better lighting for the photo, clearer photos, less foods overlapping, etc.
So in order to try and investigate this issue further the model was to be ran , still with the input image size of 512x512, but now with the model only being trained on those 10 classes to see that if the supposed interference was removed how would the model behave.
The results from this processes being ran on the model are shown below:

	TP	TN	FP	FN	Sens	Spec	Prec	IoU	Dice	Acc
bread	7466	245328	4640	4710	0.605	0.981	0.349	0.233	0.269	0.964
carrot	4518	255384	1542	700	0.837	0.994	0.473	0.408	0.444	0.991
chicken-duck	5722	245124	6016	5282	0.462	0.976	0.302	0.169	0.196	0.957
sauce	8	254266	3287	4582	0.002	0.987	0.014	0.001	0.001	0.970
tomato	13	251145	7403	3584	0.004	0.971	0.003	0.001	0.001	0.958
potato	17	250599	5427	6101	0.003	0.978	0.001	0.000	0.001	0.956
steak	26	249814	4096	8208	0.003	0.984	0.014	0.001	0.002	0.953
broccoli	0	257237	0	4907	0.000	1.000	n/a	0.000	0.000	0.981
ice-cream	0	258139	0	4005	0.000	1.000	n/a	0.000	0.000	0.985
cilantro-mint	10	256591	2707	2836	0.007	0.990	0.015	0.002	0.003	0.979
rice	0	258024	0	4120	0.000	1.000	n/a	0.000	0.000	0.984
pork	0	256482	0	5662	0.000	1.000	n/a	0.000	0.000	0.978
lemon	0	260229	0	1915	0.000	1.000	n/a	0.000	0.000	0.993
lettuce	0	259468	0	2676	0.000	1.000	n/a	0.000	0.000	0.990
strawberry	0	259912	0	2232	0.000	1.000	n/a	0.000	0.000	0.991
pie	0	257798	0	4346	0.000	1.000	n/a	0.000	0.000	0.983
cucumber	0	260044	0	2100	0.000	1.000	n/a	0.000	0.000	0.992
onion	0	260536	0	1608	0.000	1.000	n/a	0.000	0.000	0.994
corn	0	259999	0	2145	0.000	1.000	n/a	0.000	0.000	0.992
cake	0	257448	0	4696	0.000	1.000	n/a	0.000	0.000	0.982
pepper	0	260552	0	1592	0.000	1.000	n/a	0.000	0.000	0.994
cheese-butter	0	260719	0	1425	0.000	1.000	n/a	0.000	0.000	0.995
french-beans	0	260478	0	1666	0.000	1.000	n/a	0.000	0.000	0.994
fish	0	259159	0	2985	0.000	1.000	n/a	0.000	0.000	0.989
biscuit	0	259838	0	2306	0.000	1.000	n/a	0.000	0.000	0.991
egg	0	260348	0	1796	0.000	1.000	n/a	0.000	0.000	0.993
asparagus	0	260392	0	1752	0.000	1.000	n/a	0.000	0.000	0.993
noodles	0	260154	0	1990	0.000	1.000	n/a	0.000	0.000	0.992
orange	0	261289	0	855	0.000	1.000	n/a	0.000	0.000	0.997
sausage	0	260985	0	1159	0.000	1.000	n/a	0.000	0.000	0.996
french-fries	0	260441	0	1703	0.000	1.000	n/a	0.000	0.000	0.994

As seen in the results table above, when the model was traine on just those 10 classes those being:

- Bread
- Carrot
- Chicken-duck

- Tomato
- Sauce
- Potato
- Steak
- Cilantro Mint

It would still, despite being trained on less classes, diverted most of its training and learning to a third of those classes, namely it was the 3 most common classes meaning they had the largest number of images within the overall dataset these being:

- Bread - 1405 images
- Carrot - 1279 images
- Chicken-duck - 1242 images

It is unknown why the model behaved in this way and it was outside the scope of the project to be able to allocate the time needed in order to further investigate and solve this issue, but it does show promise. Despite the model only being able to train itself on a third of the dataset it would seem it still shows that of the classes it does train itself on it is capable of producing a better score for example from the 10 class train dataset for the model when it was run it produced an IoU of 0.444 with an accuracy of 0.991 for carrots which isn't bad considering that models such as FoodSAM which reports IoUs of 46.4

One potential reason that the model did worse for the 10 classes when trained on just those and performed better on the top 3 most common; bread, carrot, and chicken-duck could be caused by the lack of adversarial training in the model that was trained on just 10 classes. Seeing as it was not trained on the entire dataset with all its classes it could negatively effect the results of the model because of the lack of other classes as in , when lookin at the model that was trained on all 32 classes and only managed to segmented 10 of them, it may have gotten a wider array of scores because since it was trained on a larger number of classes, 32 but still not all 103, it was able to distinguish which food was which better than the 10 class model because due to having 32 classes it was able to determine what foods the object(s) it was looking at was NOT allowing it to effectively narrow down the number of possibilities that it could be

In order to determine if the model was just biased to these 10 most common classes the model was next ran and tested with the next 10 most common classes after the first 10 most common in order to see how would the model behave and would it still appear to learn and have better IoU and accuracy scores for the top 3 most common classes of that 10.

	TP	TN	FP	FN	Sens	Spec	Prec	IoU	Dice	Acc
bread	15	246571	3398	12160	0.001	0.986	0.016	0.000	0.001	0.941
carrot	12	250742	6184	5206	0.003	0.975	0.004	0.001	0.001	0.957
chicken-duck	116	245571	5569	10887	0.010	0.977	0.020	0.003	0.006	0.937
sauce	0	257553	0	4591	0.000	1.000	n/a	0.000	0.000	0.982
tomato	8	256837	1711	3588	0.006	0.993	0.003	0.001	0.002	0.980
potato	73	251743	4283	6045	0.011	0.984	0.022	0.004	0.006	0.961
steak	0	253910	0	8234	0.000	1.000	n/a	0.000	0.000	0.969
broccoli	0	257237	0	4907	0.000	1.000	n/a	0.000	0.000	0.981
ice-cream	4	255291	2848	4001	0.003	0.989	0.004	0.000	0.000	0.974
cilantro-mint	0	259298	0	2846	0.000	1.000	n/a	0.000	0.000	0.989
rice	0	258024	0	4120	0.000	1.000	n/a	0.000	0.000	0.984
pork	0	256482	0	5662	0.000	1.000	n/a	0.000	0.000	0.978
lemon	0	260229	0	1915	0.000	1.000	n/a	0.000	0.000	0.993
lettuce	0	259468	0	2676	0.000	1.000	n/a	0.000	0.000	0.990
strawberry	0	259912	0	2232	0.000	1.000	n/a	0.000	0.000	0.991
pie	0	257798	0	4346	0.000	1.000	n/a	0.000	0.000	0.983
cucumber	0	260044	0	2100	0.000	1.000	n/a	0.000	0.000	0.992
onion	0	260536	0	1608	0.000	1.000	n/a	0.000	0.000	0.994
corn	0	259999	0	2145	0.000	1.000	n/a	0.000	0.000	0.992
cake	0	257448	0	4696	0.000	1.000	n/a	0.000	0.000	0.982
pepper	0	260552	0	1592	0.000	1.000	n/a	0.000	0.000	0.994
cheese-butter	0	260719	0	1425	0.000	1.000	n/a	0.000	0.000	0.995
french-beans	0	260478	0	1666	0.000	1.000	n/a	0.000	0.000	0.994
fish	0	259159	0	2985	0.000	1.000	n/a	0.000	0.000	0.989
biscuit	0	259838	0	2306	0.000	1.000	n/a	0.000	0.000	0.991
egg	0	260348	0	1796	0.000	1.000	n/a	0.000	0.000	0.993
asparagus	0	260392	0	1752	0.000	1.000	n/a	0.000	0.000	0.993
noodles	0	260154	0	1990	0.000	1.000	n/a	0.000	0.000	0.992
orange	0	261289	0	855	0.000	1.000	n/a	0.000	0.000	0.997
sausage	0	260985	0	1159	0.000	1.000	n/a	0.000	0.000	0.996
french-fries	0	260441	0	1703	0.000	1.000	n/a	0.000	0.000	0.994

As seen in the table of results above despite being trained on the 11th-20th most common classes (inclusive of the 11th and 20th) it still only produced results for the most common 1st-10th class which was very odd behaviour. For reasons that were unknown it would appear that the model was struggling with the classification section of the task and was still miss classifying almost every class as the wrong class of food with achieving negligible IoU scores of 0.004 for potato and 0.001 for bread and carrot. The cause of this was unknown and would be out of the scope of this project in the given time frame to be able to investigate as needed

to find and solve the underlying issue causing this error.
Despite that being said with a few alterations and re-running the model again it was very suggestive that the segmentation of the foods wasn't as much as an issue but rather the underling issue was inside of how the model was labelling the food(s).

	TP	TN	FP	FN	Sens	Spec	Prec	IoU	Dice	Acc
background	139062	63918	49523	9640	0.938	0.537	0.730	0.693	0.802	0.774
food	63918	139062	9640	49523	0.537	0.938	0.820	0.490	0.610	0.774

This table of results above shows when all food categories were considered as one and then model was simply trained to separate food from the background in the images and in doing so for the food it was resulting in an IoU of 0.490 and a Dice score of 0.610 and these scores, although could be improved further, are considerably higher than when viewing the IoU and Dice scores for each invidual class.
This would suggest that the model rather struggles more with labelling than segmentation of the food(s). This was further reinforced to be a large contributing factor to the lower than expected IoU scores for each class when the results from the most common 10 classes and 1th-20th most common classes were adjusted, by this the outputting masks were adjusted to be set to the correct labels from the ground truth images in order to remove the labelling aspect from the segmentation model to better evaluate purely how good or bad the segmentation's were.

	TP	TN	FP	FN	Sens	Spec	Prec	IoU	Dice	Acc
bread	7466	245328	4640	4710	0.605	0.981	0.349	0.233	0.269	0.964
carrot	4518	255384	1542	700	0.837	0.994	0.473	0.408	0.444	0.991
chicken-duck	5722	245124	6016	5282	0.462	0.976	0.302	0.169	0.196	0.957
sauce	0	257553	0	4591	0.000	1.000	n/a	0.000	0.000	0.982
tomato	2481	257733	815	1116	0.685	0.997	0.428	0.306	0.348	0.993
potato	0	256026	0	6118	0.000	1.000	n/a	0.000	0.000	0.977
steak	4078	250572	3338	4156	0.480	0.987	0.302	0.176	0.206	0.971
broccoli	4524	256318	919	383	0.893	0.996	0.547	0.495	0.532	0.995
ice-cream	2095	256112	2027	1910	0.472	0.992	0.263	0.167	0.192	0.985
cilantro-mint	0	259298	0	2846	0.000	1.000	n/a	0.000	0.000	0.989
rice	0	258024	0	4120	0.000	1.000	n/a	0.000	0.000	0.984
pork	0	256482	0	5662	0.000	1.000	n/a	0.000	0.000	0.978
lemon	0	260229	0	1915	0.000	1.000	n/a	0.000	0.000	0.993
lettuce	0	259468	0	2676	0.000	1.000	n/a	0.000	0.000	0.990
strawberry	1943	259138	774	289	0.866	0.997	0.384	0.337	0.366	0.996
pie	0	257798	0	4346	0.000	1.000	n/a	0.000	0.000	0.983
cucumber	0	260044	0	2100	0.000	1.000	n/a	0.000	0.000	0.992
onion	0	260536	0	1608	0.000	1.000	n/a	0.000	0.000	0.994
corn	0	259999	0	2145	0.000	1.000	n/a	0.000	0.000	0.992
cake	0	257448	0	4696	0.000	1.000	n/a	0.000	0.000	0.982
pepper	0	260552	0	1592	0.000	1.000	n/a	0.000	0.000	0.994
cheese-butter	0	260719	0	1425	0.000	1.000	n/a	0.000	0.000	0.995
french-beans	0	260478	0	1666	0.000	1.000	n/a	0.000	0.000	0.994
fish	0	259159	0	2985	0.000	1.000	n/a	0.000	0.000	0.989
biscuit	0	259838	0	2306	0.000	1.000	n/a	0.000	0.000	0.991
egg	0	260348	0	1796	0.000	1.000	n/a	0.000	0.000	0.993
asparagus	0	260392	0	1752	0.000	1.000	n/a	0.000	0.000	0.993
noodles	0	260154	0	1990	0.000	1.000	n/a	0.000	0.000	0.992
orange	0	261289	0	855	0.000	1.000	n/a	0.000	0.000	0.997
sausage	0	260985	0	1159	0.000	1.000	n/a	0.000	0.000	0.996
french-fries	0	260441	0	1703	0.000	1.000	n/a	0.000	0.000	0.994

As can be seen in the table above for the top 10 most common classes the IoUs were drastically improved. For example strawberry previously when trained on the top 10 classes had an IoU of 0 and the model could not segment it or perhaps label it correctly but when removing the labelling task from the model it was able to produce an IoU of 0.337 and a dice of 0.366 for strawberry. Similarly broccoli also went from not being segmented or rather labelled at all to an IoU of 0.495 and a dice of 0.532. Thus further suggesting that the model struggles to label the correct items rather than segment them as much , although the

segmentation's could still further be improved.

To further illustrate this and something that would further suggest this to be true would be the image seen below:

Figure 4.1: Image from FoodSeg103 dataset showing example where segmentation was not far from being correct but the labelling was incorrect causing a lower IoU score.

In the image above looking specifically at what appears to be breaded meat. In the dataset this was labelled as fish but the segmentation model had predicted it to be bread, now as the outside of the fish is breaded this is a reasonable assumption the model could have made and is something that may be considered as a limitation since seeing as the surface of the meet is bread it would be fair for the model to assume that that is indeed bread. Further implying that its not the segmentation causing the majority of issues, although it could still be improved, but rather the labelling of segmented objects that is.

4.0.5 Volume and Depth Estimation

Once the model had been trained for segmenting the food it was time to move onto utilizing a model that would be able to predict the total volume for the food(s) in the image. It would be far outside the scope of this project to build a Volume Estimator for foods from the ground up so in order to allow the model to produce a final answer still within this project a pre-trained pre-made volume estimator would be implemented. The model that was chosen was FVE (Food Volume Estimator) [71]. This FVE model was ran with the total dataset as an input and produces segmentation masks for each of the foods in the image and a depth map for the entire image as can be seen below:

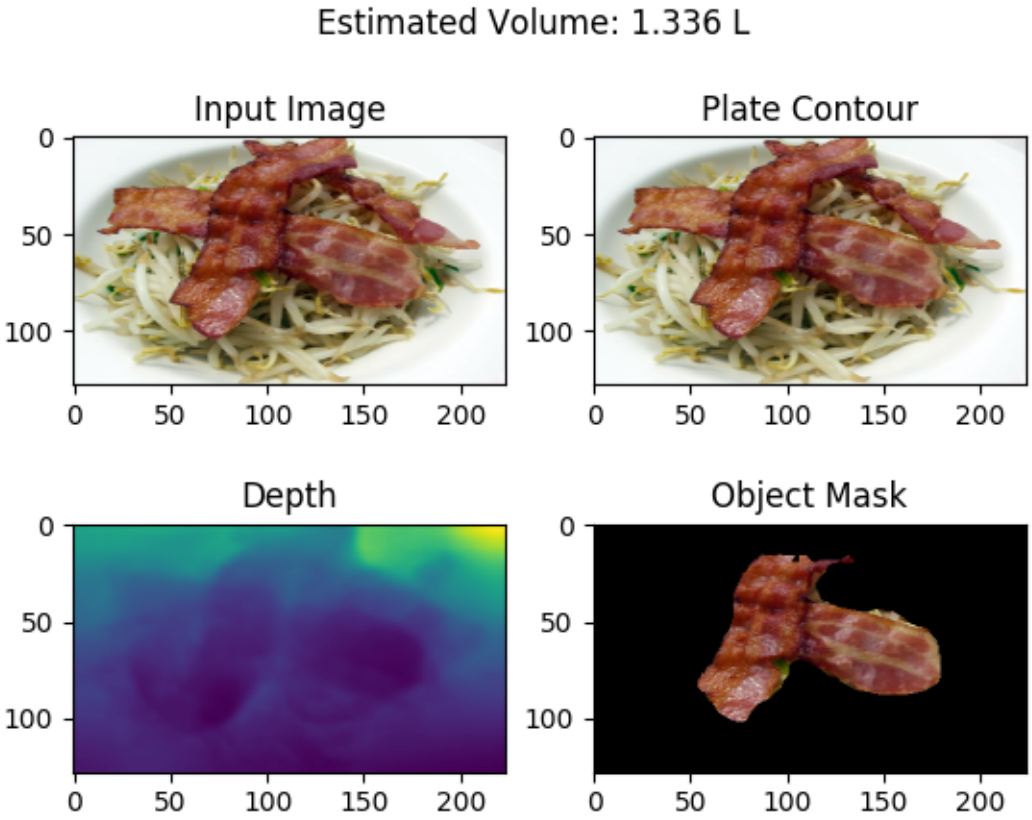


Figure 4.2: Example output for FVE

This is an example output for an input image for the FVE. There is multiple of these images outputted 1 for each segmentation produced from the image of the food(s).
Along side these images a volume in Litres is produced for each segmentation.

Figure 4.3: An image from the foodseg103 dataset that was tested to find its estimated caloric value

Taking the above image as an example the calories were calculated by firstly totalling the volume for each class of food. So in the case of the image above the totaled volume for each class was 1.105937814119939 as the FVE model only segmented 1 class, the bread. The average density for bread was taken as [29] 0.29gml so we times by 1000 to get grams per Litre since the volume is estimated in Litres. Then using the below equation

$$Estimatedweight = EstimatedVolume \times AverageDensity \tag{4.1}$$

The estimated weight of bread in this image was

$$Estimatedweight = 1.105937814119939 \times 290 \tag{4.2}$$

Equalling to 320.7219660947823 grams. This was then taken and multiplied by the calories per 100g for white bread which was 238 calories taken from the calorie able [30] in order to then calculate the estimated calories as:

$$EstimatedCalories = 763.3182793055819 = \frac{320.7219660947823}{100} \times 238 \tag{4.3}$$

Considering that the image appears to only contain 1 piece of white bread toasted and buttered. If you were to take an example 1 slice of white toast [72] which is 77 calories this can be assumed to be quite far from correct.

Something that should be noted is that it was not feasible to be able to also produced a calculation for the Macro nutrient values. This was because within the multiple datasets that were being read from in order to estimate calories there wasn't found to be a dataset contained the same foods with macro nutrient values, by this I mean that the food in question to be estimated would have to firstly be in the labels from the foodseg103 dataset so that the model could label the food, assuming its labelled correctly, secondly it would need to be in the density table to estimate a volume, and finally also be contained in a calorie table with macro nutrients in order to predict both calories and macro nutrients.

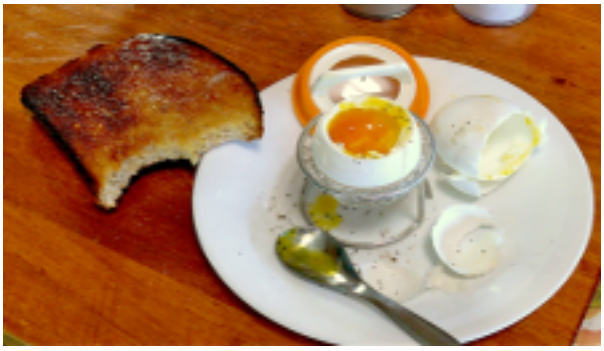


Figure 4.4: Image from the FoodSeg103 dataset showing some portion of a piece of toast and an open egg which appears to be soft boiled

Taking a look at another example image seen above in this case the methodology predicted the calories total to be 769.9366497384827 calories. Again this would appear to be incorrect as the calories for a slice of bread is 77 and this is half eaten and the calories for 1 egg is 72 calories [73] estimating this image to contain around 200 calories of food accounting some for the potential butter on the toast.

Chapter 5

Discussion

5.0.1 Overview:

When starting out the project the goal was to create a AI model that would use semantic segmentation in order to develop a methodology that would be able to , from a single image of a plate of food taken on a phone, be able to successfully estimate the caloric as well as macro-nutrient values of the food(s) presented to the model in the image. This would help people who wanted to follow a diet, to lose or gain weight, by making the process of counting calories easier, faster, and less time consuming.

5.0.2 Limitations

As previously mentioned the model has some inherent limitations which were out of the scope of this project for example the 'chicken and rice problem' or the issue of if there is no plate then the model may struggle as it can not find any ellipse, ellipse meaning the edges of the plate and or bowl, and also the problem of detecting how the food(s) were cooked, e.g. was it cooked in butter or oil.
These limitations as well as all the others mentioned within the introduction: Scope and Limitations, were not resolved over the span of this project as they were either simply inherently not possible to solve within the scope of this project such as the 'chicken and rice problem' or they were out of the scope of the project such as the FoodVolumeEstimator having to find a number of points on the edge of the plate, typically at least 5, a feasible solution to this problem but something that was out the scope of this project would be to develop a stand alone plate segmentation model that would be able to recognise the type of bowl and or plate that is is seeing within the image to better estimate the total volume that that type of bowl and or plate is able to hold within / upon it.

This would overall improve accuracy and allow for a better estimate of the caloric value of the foods. Mention no one knows what the companies doing Another limitation that was

discovered during the course of this project was that already existing approaches such as MyFitnessPal's Meal-Scan Feature [74] are behind a pay wall in order to use and regardless of this are not open source due to it being a product made for profit.

5.0.3 Further improvements and research

From the beginning of this project there were certain elements that were known to be limiting factors or rather areas of work that were out of the scope of the project and would need more time in order to develop further to be of a higher standard. One of these limitations that was known from the very beginning of this project that would need further research in order to be developed into a deploy-able state, to mean something that would be of a good enough quality to be used by users day to day would be the depth estimation / volume estimation.

Another area of the model that should be considered is the average density of foods table that is being used in combination with the estimated volume in order to be able to estimate the total weight of the food to estimate the overall calories and one way this could be improved would be through creating a specific density table for the model and the foods in the model OR also verifying the densities of the foods manually in order to confirm the densities are correct and also to add any to the table that are missing from the table that we need from our 103 classes of foods.

One other potential avenue for further research that could be looked into that could improve the model immensely would be by implementing a hierarchical model design. By this what is meant is there would be a segmentation model designed for each category of food for example a segmentation model for breads, meats, fruits, etc.
This would allow the model to have more detailed segmentation having different models with greater specificity that, with the use of an over arching model, would be combined to create a much more specific model that would be able to ideally segment different foods with a greater accuracy.
This is something though that would be outside the scope of this project as this would take a mass amount of processing power and time, as each model would have to be trained and tested on its own data set, also each dataset for each model would have to be large enough that the model would be able to learn enough from the training data.
This currently may not be feasible as it may require far too much memory and even if it didn't require too much memory to train and produce it wouldn't be practical enough for a mobile phone app to be deployed to users on the app store as it would still require a large amount of memory to run each time on the users phone.

This is something that in the future may not be an issue since as time passes and we advance technology further the 'price' of memory decreases, by this I mean what is considered a

'Large' amount of memory decreases as we improve technology almost a sort of memory inflation. For example the game "Call of Duty: Black Ops 2" was released in 2012 and was just 16GB large where as the most recent Call of Duty game to release, Modern Warfare 3 remastered, is a whopping 172GB for a PC and 240gb for playstation 5.
[75] [76] [77] Also the price of memory has decreased though for example: In January 2000 it was 1.56 dollars per Megabyte where as in January 2023 it was 0.0020 dollars per Megabyte. [78]

Another area of research that if it was further improved would considerably improve the overall performance of the segmentation model and therefore the model overall would be to solve the limitation of the segmentation model only being able to segment 10 classes. The reason the model behaves in this way was not clear during the course of the project. There were a few possible reasons that were considered and in some cases proven to NOT be the cause for example, when the model was running on 256x256 and condensing all the images every image was being padded via the Neuman boundary conditions.
Before re-running the model with images greater than the minimum size for the model, i.e. then there is no need for padding, it was thought to potentially be caused by those 10 classes containing a large amount of images where the foods of interest are nearer the edges of the image and so when the padding was added via Neuman boundary conditions those foods would then have occupy a larger proportion of the new padded image compared to how much they occupied in the original image prior to any padding being added. Once the model had been re-ran with the now larger images without any padding it was found to not be the cause as the model still only segmented 10 classes.

As discussed in results one limitation that the model had ran into was the fact that it seemed to be more limited by the labelling of objects rather than the segmentation of them. When the labelling task was removed from the model and fixed to be correct using the ground truth then the models IoU and Dice values increased drastically implying that , although the segmentation could still be improved, it was the miss classification of foods that were having more of a negative impact on the IoU and Dice scores.
One final limitation of the methodology was the at when implementing the FVE the implementation did struggle to segment the plate. The model would struggle to find the ellipses of the plate in order to be able to estimate the size of the plate which in turn when failed meant it lacked a point of reference for scale in order to estimate how large the food(s) on the plate were negatively impacting the estimated volume and therefore estimated calories.

5.0.4 Datasets

Although the dataset chosen was sufficient for this project in an ideal world we would have a perfect dataset that was crafted specifically for this model for example an ideal dataset

would have over 100,00 images, would have an even distribution of images across all classes so the model would be trained evenly over all classes.
Ideally the images would all be taken from the same angle for example top down, this may now have the constraint that the input images for the model may have to be take from the same top down angle but if this allows the model to be more consistent and more accurate then overall i would consider this to be a net positive as it will improve the accuracy of the model.
Additionally the model would perform better if it had had been trained on multiple data sets but this was not feasible within this project. If it was the dataset would have a much greater variety within the angle and lighting of the photos taken, the foods in the dataset, etc. This would in theory allow the model to have greater generalizability and therefore improving its performance.

Chapter 6

Conclusions

6.1 Overview

When starting out this project the aim was to produce a model that would be able to be used to estimate the calories on a plate of food with the goal of allowing people to easily track their calories in order to lose weight and or lead a healthier lifestyle through their diet. The approach that was used in order to do this was to create a semantic segmentation architecture that would be able to recognise the boundaries of the food(s) on the plate of food, once this was done then using a pre-trained depth estimation model to estimate the volume of food that was being shown in the image, then the model would use the result of this model in coordination with the data from a table of average density of different foods to then obtain an estimated value for the total weight of food we were viewing in the image, thus allowing us to estimate the total caloric value of all of the food(s) in the image.

6.2 Evaluating the Model

Upon viewing the model to evaluate its performance as a whole considering the goal that was set at the start of this project it is fair to say that the model would need more work in order to be up to par with current models such as FoodSAM. FoodSAM had an average IoU of 0.46cite and some of this models results for certain classes were close to if not above 0.46 for example with 32 classes once the labellings were appended scoring an IoU of 0.813 for corn! Overall thought without the label classificatios being made the models performance was much poorer on that same set without the adjustments scoring a 0 for corn not being able to correctly label it at all. In order to be deployed and useable in the real world this model would need more work in the segmentation model and the volume estimation model.

6.3 Further work and Improvements

Upon reflection and careful consideration of all the work and the final version of the model that was produced it can be said that to be deployed into an app format so that it could be used day to day by users it would need more work to be reliable enough. Fundamentally the biggest area for improvement would be the IoU of the segmentation model. It was discovered that through testing the model with different subsets of the FoodSeg103 dataset the model seemed to perform noticeably better at segmenting the objects than labelled the as it struggled with correctly labelling the objects as the correct class.
If this area of the model was improved it would improve the estimated calorie output massively and bring it closer to being a fully deploy-able application. It would also be important for the depth estimation to be improved and perhaps developed specifically for this methodology. Although creating the methodology's own FoodVolumeEstimator would improve the model's performance considerably, I think the segmentation would still be of a higher priority as if the model cant segment the food with a high level of accuracy, IoU and Dice then it can not be effective at estimating the caloric value of the food it is shown despite however accurate the depth estimation may be.

6.4 Personal reflection

Throughout this project my own personal knowledge of AI, machine learning and deep learning have expanded drastically. I've learnt how models are trained and tested, how a model should be optimised for the solution at hand, and how to recognise a suitable model for a given problem. This project has definitely peaked my interest in AI as a whole and has allowed me to realise and think more creatively about possible solutions that are able to be built due to AI technologies. It has also shown me though some of the challenges current research faces in AI and some of the complexities that come with building a deep learning model.

References

[1] "Obesity: The story so far," *BBC News*, Aug. 25, 2011. [Online]. Available: https://www.bbc.com/news/health-14669209.

[2] "Obesity and overweight." (), [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight.

[3] "Obesity," nhs.uk. Section: conditions. (Nov. 23, 2017), [Online]. Available: https://www.nhs.uk/conditions/obesity/.

[4] "Obesity and overweight." [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight.

[5] "What is a calorie deficit, and how much of one is healthy?" Healthline. (Dec. 12, 2019), [Online]. Available: https://www.healthline.com/nutrition/calorie-deficit.

[6] "Where does fat go when you lose weight?" Healthline. (Apr. 8, 2020), [Online]. Available: https://www.healthline.com/nutrition/where-does-fat-go-when-you-lose-weight.

[7] "How fat affects your GI tract." (), [Online]. Available: https://www.orlandohealth.com/content-hub/how-fat-affects-your-gi-tract.

[8] "Is a low-fat diet healthy?" Good Food. (Aug. 1, 2020), [Online]. Available: https://www.bbcgoodfood.com/howto/guide/spotlight-low-fat-diets.

[9] S. A. Bilsborough and T. C. Crowe, "Low-carbohydrate diets: What are the potential short- and long-term health implications?" *Asia Pacific Journal of Clinical Nutrition*, vol. 12, no. 4, pp. 396–404, 2003, ISSN: 0964-7058.

[10] "Low carb diets often influenced by internet information and do not work for all." (), [Online]. Available: https://www.gla.ac.uk/news/archiveofnews/2020/september/headline_738322_en.html.

[11] A. Astrup, "The satiating power of protein—a key to obesity prevention?" *The American Journal of Clinical Nutrition*, vol. 82, no. 1, pp. 1–2, Jul. 1, 2005, Publisher: Elsevier, ISSN: 0002-9165, 1938-3207. DOI: 10.1093/ajcn/82.1.1. [Online]. Available: https://ajcn.nutrition.org/article/S0002-9165(23)29500-3/fulltext.

[12] J. A. Levine, "Non-exercise activity thermogenesis (NEAT)," *Best Practice & Research. Clinical Endocrinology & Metabolism*, vol. 16, no. 4, pp. 679–702, Dec. 2002, ISSN: 1521-690X. DOI: 10.1053/beem.2002.0227.

[13] "Pizza customization — domino's pizza." (), [Online]. Available: https://www.dominos.co.uk/store/28394/lancaster/menu/pizza/15/54/Mighty%20Meaty%C2%AE.

[14] "Tools - calorie calculator - calorie calculator." (), [Online]. Available: https://www.runnerspace.com/gprofile.php?mgroup_id=45577.

[15] *Half marathon world record progression*, in *Wikipedia*, Page Version ID: 1181482626, Oct. 23, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Half_marathon_world_record_progression&oldid=1181482626.

[16] "Food — the water witch — lancaster," LIVESite Water Witch. (), [Online]. Available: https://www.waterwitchlancaster.co.uk/food.

[17] S. W. Lichtman, K. Pisarska, E. R. Berman, *et al.*, "Discrepancy between self-reported and actual caloric intake and exercise in obese subjects," *The New England Journal of Medicine*, vol. 327, no. 27, pp. 1893–1898, Dec. 31, 1992, ISSN: 0028-4793. DOI: 10.1056/NEJM199212313272701.

[18] "Brits lose count of their calories: Over a third of brits don't know how many calories they consume on a typical day." Section: Uncategorised. (), [Online]. Available: https://www.mintel.com/press-centre/brits-lose-count-of-their-calories-over-a-third-of-brits-dont-know-how-many-calories-they-consume-on-a-typical-day/.

[19] "14.9. semantic segmentation and the dataset — dive into deep learning 1.0.3 documentation." (), [Online]. Available: https://d2l.ai/chapter_computer-vision/semantic-segmentation-and-dataset.html.

[20] "Deep learning." (), [Online]. Available: https://www.deeplearningbook.org/.

[21] "Figure 1. schematic diagram of a basic convolutional neural network...," ResearchGate. (), [Online]. Available: https://www.researchgate.net/figure/Schematic-diagram-of-a-basic-convolutional-neural-network-CNN-architecture-26_fig1_336805909.

[22] "CNN — introduction to pooling layer," GeeksforGeeks. Section: AI-ML-DS. (Aug. 5, 2019), [Online]. Available: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/.

[23] "Classification in machine learning: A guide for beginners." (), [Online]. Available: https://www.datacamp.com/blog/classification-machine-learning.

[24] "Calculation intersection over union (IoU) for evaluating an image segmentation model," GeeksforGeeks. Section: Java. (Mar. 3, 2023), [Online]. Available: https://www.geeksforgeeks.org/calculation-intersection-over-union-iou-for-evaluating-an-image-segmentation-model-using-java/.

[25] "Epoch in neural networks — baeldung on computer science." (Dec. 29, 2020), [Online]. Available: https://www.baeldung.com/cs/epoch-neural-networks.

[26] *Sørensen–dice coefficient*, in *Wikipedia*, Page Version ID: 1193861550, Jan. 6, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=S%C3%B8rensen%E2%80%93Dice_coefficient&oldid=1193861550.

[27] "How to calculate dice probabilities?" GeeksforGeeks. Section: Mathematics. (Dec. 30, 2021), [Online]. Available: https://www.geeksforgeeks.org/how-to-calculate-dice-probabilities/.

[28] N. O.
bibinitperiod A. A. US Department of Commerce. "What is LIDAR." (), [Online]. Available: https://oceanservice.noaa.gov/facts/lidar.html.

[29] U. R. Charrondiere, D. Haytowitz, and B. Stadlmayr, "FAO/INFOODS density database version 1.0 (july 2011),"

[30] "Calories in food items (per 100 grams)." (), [Online]. Available: https://www.kaggle.com/datasets/kkhandekar/calories-in-food-items-per-100-grams?resource=download.

[31] "What is the body mass index (BMI)?" nhs.uk. Section: chq. (Jun. 26, 2018), [Online]. Available: https://www.nhs.uk/common-health-questions/lifestyle/what-is-the-body-mass-index-bmi/.

[32] "Obesity prevalence adults by country 2021 — statista." (), [Online]. Available: https://www.statista.com/statistics/236823/prevalence-of-obesity-among-adults-by-country/.

[33] "Prevalence of obesity among population aged 2–19 years u.s. 1971-2018," Statista. (), [Online]. Available: https://www.statista.com/statistics/1242081/prevalence-of-obesity-among-us-children-and-adolescents/.

[34] "Diabetes," nhs.uk. Section: conditions. (Oct. 18, 2017), [Online]. Available: https://www.nhs.uk/conditions/diabetes/.

[35] "TDEE calculator: Learn your total daily energy expenditure." (), [Online]. Available: https://tdeecalculator.net/.

[36] "Use the health app – apple support (UK)," Apple Support. (), [Online]. Available: https://support.apple.com/en-gb/104997.

[37] "MyFitnessPal revenue and usage statistics in 2023," Dev Technosys. (), [Online]. Available: https://devtechnosys.com/data/myfitnesspal-statistics.php.

[38] "Meal scan FAQ," MyFitnessPal Help. (), [Online]. Available: https://support.myfitnesspal.com/hc/en-us/articles/360045761612-Meal-Scan-FAQ.

[39] "Best curried chicken fried rice with chilies recipe - how to make curried chicken fried rice with chilies." (), [Online]. Available: https://www.177milkstreet.com/recipes/curried-chicken-fried-rice-with-chilies.

[40] "Butter vs. olive oil nutrition," *The Times of India*, ISSN: 0971-8257. [Online]. Available: https://timesofindia.indiatimes.com/life-style/food-news/butter-or-olive-oil-the-best-fat-to-cook-an-omelette-to-get-the-maximum-benefits-out-of-it/photostory/89831167.cms?picid=89831181.

[41] "Tesco 2 british chicken breast fillets 300g." (), [Online]. Available: https://www.tesco.com/groceries/en-GB/products/285210252.

[42] "Tesco british pork sausages 8 pack 454g." (), [Online]. Available: https://www.tesco.com/groceries/en-GB/products/261879050.

[43] N. Otsu *et al.*, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.

[44] "Backpropagation in machine learning," GeeksforGeeks. Section: Machine Learning. (Mar. 4, 2024), [Online]. Available: https://www.geeksforgeeks.org/backpropagation-in-machine-learning/.

[45] "Activation functions in neural networks — by SAGAR SHARMA — towards data science." (), [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

[46] "Edge detection using prewitt, scharr and sobel operator," GeeksforGeeks. Section: MATLAB. (Oct. 11, 2021), [Online]. Available: https://www.geeksforgeeks.org/edge-detection-using-prewitt-scharr-and-sobel-operator/.

[47] "Feature detectors - sobel edge detector." (), [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm.

[48] "Masks for the prewitt gradient edge detector the laplacian operator is... — download scientific diagram." (), [Online]. Available: https://www.researchgate.net/figure/Masks-for-the-Prewitt-gradient-edge-detector-The-Laplacian-operator-is-based-on-second_fig3_317754223.

[49] D. Mumford and J. Shah, "Optimal approximations by piecewise smooth functions and associated variational problems," *Communications on Pure and Applied Mathematics*, vol. 42, no. 5, pp. 577–685, Jul. 1989, ISSN: 0010-3640, 1097-0312. DOI: 10.1002/cpa.3160420503. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/cpa.3160420503.

[50] T. Chan and L. Vese, "Active contours without edges," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 266–277, Feb. 2001, ISSN: 10577149. DOI: 10.1109/83.902291. [Online]. Available: http://ieeexplore.ieee.org/document/902291/.

[51] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation," arXiv.org. (May 18, 2015), [Online]. Available: https://arxiv.org/abs/1505.04597v1.

[52] "U-net architecture explained - GeeksforGeeks." (), [Online]. Available: https://www.geeksforgeeks.org/u-net-architecture-explained/.

[53] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, *UNet++: A nested u-net architecture for medical image segmentation*, version: 1, Jul. 18, 2018. DOI: 10.48550/arXiv.1807.10165. arXiv: 1807.10165[cs,eess,stat]. [Online]. Available: http://arxiv.org/abs/1807.10165.

[54] "Unet++ architecture explained," GeeksforGeeks. Section: Machine Learning. (Jul. 31, 2023), [Online]. Available: https://www.geeksforgeeks.org/unet-architecture-explained/.

[55] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, *Pyramid scene parsing network*, version: 2, Apr. 27, 2017. DOI: 10.48550/arXiv.1612.01105. arXiv: 1612.01105[cs]. [Online]. Available: http://arxiv.org/abs/1612.01105.

[56] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, *Feature pyramid networks for object detection*, Apr. 19, 2017. DOI: 10.48550/arXiv.1612.03144. arXiv: 1612.03144[cs]. [Online]. Available: http://arxiv.org/abs/1612.03144.

[57] "FPN — CloudFactory computer vision wiki." (), [Online]. Available: https://wiki.cloudfactory.com/docs/mp-wiki/model-architectures/fpn.

[58] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, Curran Associates, Inc., 2012. [Online]. Available: https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

[59] S. Bangar. "AlexNet architecture explained," Medium. (Jun. 24, 2022), [Online]. Available: https://medium.com/@siddheshb008/alexnet-architecture-explained-b6240c528bd5.

[60] "Multi-class neural networks: Softmax — machine learning," Google for Developers. (), [Online]. Available: https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax.

[61] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, Dec. 10, 2015. DOI: 10.48550/arXiv.1512.03385. arXiv: 1512.03385[cs]. [Online]. Available: http://arxiv.org/abs/1512.03385.

[62] *Vanishing gradient problem*, in *Wikipedia*, Page Version ID: 1199982033, Jan. 28, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Vanishing_gradient_problem&oldid=1199982033.

[63] X. Lan, J. Lyu, H. Jiang, *et al.*, "FoodSAM: Any food segmentation," *IEEE Transactions on Multimedia*, pp. 1–14, 2024, ISSN: 1520-9210, 1941-0077. DOI: 10.1109/TMM.2023.3330047. arXiv: 2308.05938[cs]. [Online]. Available: http://arxiv.org/abs/2308.05938 (visited on 03/22/2024).

[64] S. Zheng, J. Lu, H. Zhao, *et al.*, *Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers*, version: 3, Jul. 25, 2021. arXiv: 2012.15840[cs]. [Online]. Available: http://arxiv.org/abs/2012.15840 (visited on 03/22/2024).

[65] X. Wu, X. Fu, Y. Liu, E.-P. Lim, S. C. H. Hoi, and Q. Sun, *A large-scale benchmark for food image segmentation*, version: 1, May 11, 2021. arXiv: 2105.05409[cs]. [Online]. Available: http://arxiv.org/abs/2105.05409 (visited on 03/22/2024).

[66] "Papers with code - FoodSeg103 dataset." (), [Online]. Available: https://paperswithcode.com/dataset/foodseg103.

[67] "UECFoodPix,UECFoodPixComplete." (), [Online]. Available: https://mm.cs.uec.ac.jp/uecfoodpix//#labels (visited on 03/22/2024).

[68] "Papers with code - FoodSAM: Any food segmentation." (), [Online]. Available: https://paperswithcode.com/paper/foodsam-any-food-segmentation.

[69] A. H. .-. Cheng and D. T. Cheng, "Heritage and early history of the boundary element method," *Engineering Analysis with Boundary Elements*, vol. 29, no. 3, pp. 268–302, Mar. 1, 2005, ISSN: 0955-7997. DOI: 10.1016/j.enganabound.2004.12.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0955799705000020 (visited on 03/22/2024).

[70] A. Graikos, *AlexGraikos/food_volume_estimation*, original-date: 2019-04-26T09:53:07Z, Mar. 15, 2024. [Online]. Available: https://github.com/AlexGraikos/food_volume_estimation (visited on 03/22/2024).

[71] P. Kadam, S. Pandya, S. Phansalkar, *et al.*, "FVEstimator: A novel food volume estimator wellness model for calorie measurement and healthy living," *Measurement*, vol. 198, p. 111 294, Jul. 1, 2022, ISSN: 0263-2241. DOI: 10.1016/j.measurement.2022.111294. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224122005358.

[72] "Calories in 1 slice white bread," Nutritionix. (), [Online]. Available: https://www.nutritionix.com/food/white-bread/1-slice.

[73] "Calories in egg," Nutritionix. (), [Online]. Available: https://www.nutritionix.com/food/egg.

[74] S. Sung. "Introducing the enhanced meal scan feature!" MyFitnessPal Blog. (Dec. 2, 2020), [Online]. Available: https://blog.myfitnesspal.com/meal-scan/.

[75] "Call of duty®: Black ops II on steam." (), [Online]. Available: https://store.steampowered.com/app/202970/.

[76] "Call of duty®: Modern warfare® III on steam." (), [Online]. Available: https://store.steampowered.com/app/2519060/Call_of_Duty_Modern_Warfare_III/.

[77] W. Yin-Poole. "Activision explains huge call of duty modern warfare 3 file sizes," IGN. (Nov. 2, 2023), [Online]. Available: https://www.ign.com/articles/activision-explains-huge-call-of-duty-modern-warfare-3-file-sizes.

[78] "Historical memory prices 1957+." (), [Online]. Available: https://jcmit.net/memoryprice.htm.

CALORIE ESTIMATOR

JOSHUA NOLAN, 38421682

School of Computing and Communications

Lancaster University

1. Introduction

A calorie estimator is a model that can scan a plate of food and, using a given image, estimate the total caloric value of that plate of food. One of the biggest issues people have when wanting to track their caloric intake is accuracy, especially in certain scenarios such as a restaurant. With 48% of Brits have tried to lose weight in the last year yet **well over one third (37%) of Brits don't know how many calories they consume on a typical day.** To track calories and macro nutrients accurately enough a person should ideally by weighing their food out using a scale. Calories being the unit of measure for the amount of energy a food releases when its digested in your body, and macro nutrients being what the caloric value is made up of, that being carbohydrates, fats, and proteins. These are very important in a person's diet as they are just as important as how many calories someone is eating. Now in some scenarios for example when eating a meal in a restaurant, this isn't feasible for most people. For some it may be their lifestyle if your busy and working long days you wont have the time to be weighing all your foods out let alone cooking them. Alternatively most people are more likely to lack the will power and or effort to weigh all their food out ('65% of dieters return to their pre-diet weight within 3 years'). The previously mention reasons lead to not many people , over the long term, measuring their calories accurately enough via weighing their foods out . If these barriers to tracking calories were removed it would, potentially, lead to a large increase in the number of people tracking their calories and therefore being far more likely to see results from a diet. This would presumably allow them to stick to their diet for longer as they'll start to see results faster.

There are already some attempted solutions to this issue using AI to recognize a plate of food but they lack the detail, accuracy and consistency to be viable long term for users. Most of these are applications not published as research but more released as products and so it's quite unknown how much time and resources does or doesn't go into these applications.

Image segmentation is technique used in image processing, within machine learning, which is where an image is broken down into segments, or objects, by the pixel. For example we might outline a dog in a photo, or a person, or a bike, etc. Within segmentation we have semantic segmentation, this is where different objects are outlined and also labelled, so it will distinguish between a person and a car but will also label them as a person and a car.

This differs from regression, regression is a machine learning model which compares 2 variables to estimate a numerical value, so in contracts to classification where we are trying to say if it is one category or another, so in a very binary way, regression is a lot less binary as we are giving it a numerical value to represent how the 2 variables relate to one another.

The goal of the project is to develop the methodology to estimate the number of calories of a plate of food breaking down the plate into each foods caloric value and also macro nutrients.

2. Motivation and Background

Lit Review:

Segmentation:

Image segmentation is taking an image and breaking it down into different classes or objects, i.e. being able to separate chicken from beef. As previously mentioned I have segmentation and semantic segmentation , the models I am going to talk about will be talking about both segmentation (Where objects or subjects are outlined) and semantic segmentation (where the outlined objects are labelled)

Within Image Segmentation we have different architectures we can use. One of the earliest models of involving deep learning would be Thresholding, this is where each subject is put through a logistic regression model and is give a score, between 0 and 1, then it is determined which class it should be assigned to based off of this score, for example a score of or over 0.5 may be assigned to class 2 and a score under 0.5 may be assigned to class 1. Another one of these is UNET++, which builds upon the standard UNET architecture. A **UNET** architecture essentially has an **encoder** and **decoder** which work on multiple layers (it encodes and decodes then image multiple times adding more detail with each layer). The encoder's role is to interpret the images it is given and compress the image down into features adding more features in more detail with each new layer / each new time it encodes the image. The decoder's job is to decrypt the encoded image and to produce segmentation masks from this process. The UNET++ version builds upon and improves upon the standard UNET architecture by aiming to reduce the semantic gap between each feature map at every level of the encoder and decoder. At each level in the UNET architecture the encoder and decoder don't always

produce an output that closely represents the input meaning that the decoded image at level X may not correlate to the encoded image at the same level. This is where UNET++ attempts to improve upon UNET by implementing a model where the encoded image at level X will be more closely represented when its decoded at the same stage in the decoder as it was encoded in the encoder. It does this by having its convolutional layers on skip pathways to bridge the semantic gap, a skip pathway consists of a dense convolution block with three convolution layers.

Inside of these models we have different encoders and decoders we can choose from. The ecoders role is to interpret the image data effectively to be able to learn certain features within an image, and the decoders job is to decode the encoded image to semantically extract these features onto a higher resolution image. We may consider the ImageNet encoder. This was created by Krizhevsky in 2012. The Architecture for this encoder contains eight layers; the first give layers are convolutional (meaning they take an input, apply an operation to this input and send it to the next layer) and the last 3 are fully-connected. The output produced at the end of this by the final layer is to a softmax (a softmax takes a vector of values and turns them into a vector of values that total to 1) which then produces a distribution over 1000 class labels. For layers 2,4, and 5 the layers are all connected, meaning that the neurons in one of these layers if fully connected to the neurons in the layer before it

Pyramid Scene Parsing Network (PSPNET):

This is based on semantic segmentation and is aimed at identifying specific parts of an image where the goal is to assign each pixel in the image a category label. Scene parsing provides complete understanding of the scene. It predicts the label, location, as well as shape for each element. Context is very important and allows the model to better identify what it is looking at.

For example, in *this* paper from 2017 its mentioned that before the context was given of it being a boathouse the model couldn't detect boat and assumed it to be a car but after the context was given of it being a boathouse near a river it was correctly able to recognise the object as a boat.

Pyramid Scene Parsing Networks combine the use of local knowledge like in this case along with the global cues to be able to combine them to help it better its ability to identify local objects by using the global cues as context and overall understand what the whole image is showing.

The architecture of PSPNET is that, firstly it is given a single input image. This is then passed to an encoder, for example it might be VGG as the encoder, this encoder will help extract features and encode the image. It is next passed through a Pyramid Pooling Module, what this does is it's a module for semantic segmentation and acts as global context of the image. The module can then take different sub region representations.

MOTIVATION

The issue is the most people can't be bothered to track their calories or even if you do it's impossible to track the calories of a meal when eating at a restaurant unless stated on the menu, which it rarely is. Other apps such as my fitness pal have attempted to tackle this issue with this solution, but this part of their app is behind a pay wall and from user reviews it doesn't seem to be viable day to day. Our model would aim to be usable to help make calorie tracking easier.

Another motivation is the current obesity epidemic. Although this is something most people are aware of, its something not a lot is done about. In 2020 33.1% of Men were obese while 33.9% of women were obese (having a BMI over 30). Potentially even more alarming is childhood obesity rates, in 2011-2012 16.9% of children were obese with almost 1 third being overweight or obese (31.8%). This methodology is something that when applied in this scenario could help somewhat tackle this issue through AI.

3. Aims and Objectives

Our overall aim of this project is a methodology that can estimate the total caloric value of a plate of food in a given image, within an acceptable range of accuracy, thus allowing people to more easily stick their calories and making the habit of tracking calories more viable long term.

a. Objective 1

- Firstly it is necessary to find a suitable data sets to use. Our data set will be comprised of as many photos of foods as possible, ideally from different angles with different lighting and as much variance as possible. It will also contain masks for each of the photos , preferably with different RGB values for each type of food on the plate allowing the model to understand how to recognize and separate foods from each other

b. Objective 2,…..

- Secondly Creating our approach for image segmentation. We will be starting by comparing different backbones and architectures within segmentation, and also compare different encoder and decoder combinations, for example, PSP, VGG, alexnet, resnet, and more to be able to recognise which architecture would be best suited for our methodology. We can then use this architecture to identify each of the different foods separate from one another as each food will have its own unique caloric value and macro nutrients.

c. Objective 3,…..

- We will next implement a form of depth perception to estimate the quantity of the food on the plate. We will compare different methods for this which may contain, Lidar or using

photogrammetry, to see which model works best with our data sets. Using the size of the plate we will estimate the dimensions (the length, depth, width) of each food to then calculate the caloric and nutritional values of the foods on the plate. We are going to estimate the average size of the plate to be able to scale the food relative to the size of the plate.

d. Objective 4,…..

- We will then use a large data set API of food nutrients to map the values of each food scanned. We may need to compare different APIs here to see which suits our model best, has the most data, the most accurate data, and overall is the most accurate for our goals. For example we could use MyFitnessPal's API which has over 6 million foods itesm and brands in it's database. to map the foods to find their calories and nutrients. E.G. 300g of chicken breast will be abouts 320kcal with 90g of protein.

e. Objective 5,…..

- The model will then ask the user a short set of questions to account for some small changes the camera may not be able to see, e.g. was it cooked in butter? Depending on what the users answers are it will impact how the foods value's are calculated for example cooking in butter will increase the caloric value of that food.

4. Methods

We will be using different methods of segmentation. We have the UNET architecture, UNET is a convolutional neural network which was originally designed to be used in biomedical image segmentation. It is a U shaped architecture which has an encoder (which extracts features from the input image) and a decoder (which constructs a segmentation mask using the extracted features) which are passed the given input data / image. We'll use this to distinguish each food from each other.

Within UNET we shall test and experiment with different encoders and decoders and different combinations (framework) of them to sew which ones work best for our goal and will be most efficient whilst being the most accurate.

One we may try is alexnet. It is an 8 layer deep CNN (convolutional neural network) with 5 convolutional layers and 3 fully connected layers. At the end of each layer, except for the last one, a soft max (converts a vector of X real numbers into a probability distribution) is produced

with 1000 class labels. The kernels for the 2nd,4th, and 5th layers are connected to the kernel maps in the previous layer, which are on the same GPU. The kernels of the 3rd layer are connected to all the kernels in the 2nd layer. It also uses drop outs which is when the encoder sets a zero to the output of each hidden neuron that has a probability of 0.5. Alex net also uses ReLU (Rectified Linear Unit) Nonlinearity to train its model much faster than previously This encoder aims to improve upon ImageNet and make it faster. Alexnet achieved a top 5 error competing in the 2012 ImageNet Large Scale Visual Recognition Challenge of 15.3% which was 10.8% lower than that of the runner.

We will also take into considering the overhead of different models as we may not need as many parameters and be able to get the same accuracy with less parameters whilst at the same time reducing the over head and so improving its speed.

There are other methods we can use such as transformers. Transformers look at an image more broadly than kernels. Unlike kernels which look at pixels of an image then move on, Transformers break down the image into 12 larger chunks. This can specifically be used as a form of edge detection.

Figure: Image showing segmentation on a plate of food

The above image demonstrates some current issues in the field of segmentation on specifically food. As you can see above it virtually recognises none of the edges of any of the foods correctly, even if it had accurate depth estimation and an accurate database for the foods caloric and macro nutrient values, it wouldn't be away near close to the correct output.

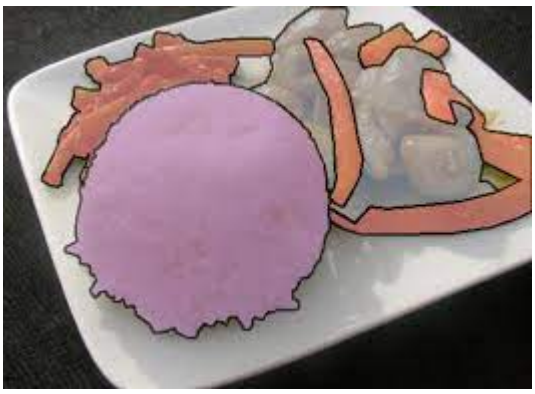


Figure: Image showing segmentation on a plate of food more accurately

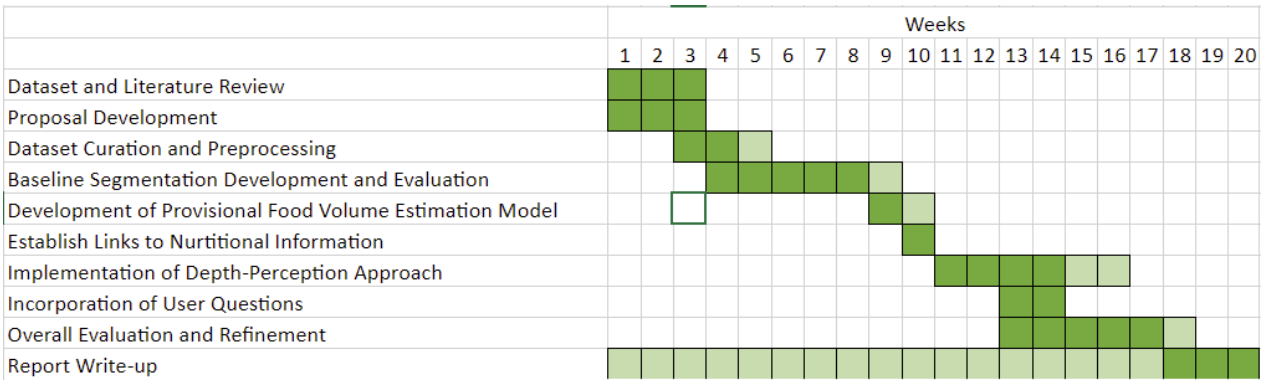
This above image shows potentially a more accurate approach to image segmentation with foods as it appears to be able to separate each food despite foods sitting on top of one another, assigning each food its own RGB value which is then mapped to a specific food to collect that foods caloric and nutritional values.

5. Data

Details and examples of the data that you will use.

6. Project Timeline

Gannt chart showing the timeline of the project.



7. References

1. “Brits lose count of their calories: Over a third of Brits don’t know how many calories they consume on a typical day. Mintel. 2016 March 9 [cited 2023 October 23]. Available from: https://www.mintel.com/press-centre/brits-lose-count-of-their-calories-over-a-third-of-brits-dont-know-how-many-calories-they-consume-on-a-typical-day/” Zhou2018Unet++.pdf
2. “33.1% of Men were obese while 33.9% of women” - https://www.statista.com/statistics/236823/prevalence-of-obesity-among-adults-by-country/
3. https://openaccess.thecvf.com/content_cvpr_2017/papers/Zhao_Pyramid_Scene_Parsing_CVPR_2017_paper.pdf
4. https://www.statista.com/statistics/505986/adult-and-childhood-obesity-share-in-the-us/
5. https://www.livestrong.com/article/13764583-diet-statistics/
6. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7641788/#:~:text=MyFitnessPal%20is%20characterized%20by%20its,smartphone%20and%20web-based%20versions.
7. https://medium.com/analytics-vidhya/concept-of-alexnet-convolutional-neural-network-6e73b4f9ee30
8. Image of segmentation used in methods - https://www.semanticscholar.org/paper/Food-image-analysis%3A-Segmentation%2C-identification-He-Xu/e0398ab99daa5236720cd1d91e5b150985aac4f3/figure/2
9. “It achieved a top-5 error of 15.3%. This was 10.8% lower than that of runner up.” - https://www.mygreatlearning.com/blog/alexnet-the-first-cnn-to-win-image-net/#:~:text=The%20figure%20shown%20below%20shows,on%20the%20CIFAR-10%20dataset.
10. 2nd Image of food segmentation: https://arxiv.org/pdf/2105.05409.pdf