Methodology

Part I: Dataset & Object Classification

After a thorough analysis on the current research in this area, focusing on the algorithms used, and different types of datasets, it was time to start thinking about what approaches was to be taken for this research. In machine learning research areas, the dataset is of utmost importance because it would determine how well the main algorithm would perform when generating results, so the first task was to obtain a good quality primary source of data. Since the research is based around Maltese food stuff, there was a high chance that a ready-made dataset could not be obtained from the Internet. With this in mind, research was still done to check for ready-made datasets, but none were found. So, the next task was to create a dataset for Maltese food items . The photos of this dataset were to be made up from a combination of six annotated Maltese foods, whilst also including objects with no annotations to serve as fake objects. This was done to increase the number of variables when it came to detecting the targeted items as the possibility of detecting a food which was not part of the research might give more insight to how well the classifier performs.As of recent years neural networks have been very much so a hot topic in the world of machine learning, especial when it came to the field of object classifications. In 2012 Krizhevsky Et. Al, released a paper which stated that performance can be greatly increased if multiple layers are added to neural networks rather than just having one. Even though neural networks had been studied since the late 1950’s, this breakthrough was only possible due to the increase in computing power and larger sets of data which became available. Nowadays many different variations of neural networks have been introduced and researched and each of which has their specific purpose. The classified which was opted for in this research was ‘Mask R-CNN’ because, from previous literature, this classifier deemed to show promising results in both object classification and instance segmentation whilst also being very simple to train. Having a classifier with instance segmentation deemed to be interesting, since the pixel-by-pixel detection would help in getting greater results in the calorie estimation portion of the research. Apart from all the benefits which was previously mentioned, whilst reading for a degree at MCAST (Malta College of Arts, Science & Technology), during my second year, some research and experimentation was done about the possibility of detecting “Pastizzi” in images whilst also using Mask R-CNN as the classifier. Due to this, familiarity with the classifier was already gained making it much easier to build on that knowledge.

After choosing the detection algorithm, it was time to plan on how the dataset would be structured and what it would be made up of. One of the most common problems which researchers faced whilst conducting similar works was volume, this is because it is extremely difficult to identify the depth of an item from a top view image alone. To identify such values would require multiple angled photos of each item being identified, which brings up further issues. In consequence of this, it was decided to find Maltese food items which would have a standard volume size throughout. This meant that, the main factor which would impact the nutritional value being estimated would be the size, excluding the volume, thus made this the independent variable of the study. Based off the information that was taken into consideration, six Maltese foods where chosen. These items where “Pastizzi”, “Qagħaq tal-Għasel”, “Zalzett Malti”, “Imqaret”, “Qassatat” and finally “Ġbejniet”. Other food items such as “Imqarrun il-Forn” or “Bragioli” were also taken into consideration but due to these items being filled with many different ingredients which varies between one household to another, it would be very difficult to predict what would be inside of them. Having decided this, it did not mean that the chosen foods do not vary, but very little varies between the same food item from one shop to another.

Another issue which many faced during their research was how could one identify the size of the classified object in real life, just from the image. This was very important to identify whilst planning the dataset as it would have determined if any object would need to have been included in the images or not. In previous litretaures, multiple different methods of how this was tackled could be found, an interesting approach was that of “AR DeepCalorieCam” (T. Ege, et. al, 2019), but unfortunately, by making use of a similar approach would mean that the software would only work on an iPhone 11 or better. Therefore, it was opted with the most common approach taken in previous research, using a reference object to identify its actual size. This technique is when an object with a standard size is detected along with the other items being identified, in this case it was opted to be a two-euro coin. It was decided to be a coin because its size is always the same, the probability of it changing in the near future is small and it can be found commonly in everyone’s pockets. The assumption that the images where all going to be taken from the top was also taken, making this our control variable. This was done because if an image is taken from the side, from the image on its own, it is not possible to identify the area of each item thus the estimated nutritional value would be completely off from the real values being gathered. Finally, the last feature which had to be decided before starting the data collection portion was, what program could be used to annotate the images and generate the required files. Since Mask-RCNN has multiple layers and can be used for both Semantic segmentation and object detection, the algorithm can use both bounding box annotations and polygon annotations, depending on which type of segmentation one would like to do. For the proposed research, it was opted provide annotations in the form of polygons to be able to conduct semantic segmentation. To do so, the program “VGG Image Annotator[[1]](#footnote-1)” or “VIA” was used as it deemed very simple to use and had the option to export the annotation files in COCO format as a JSON file, which was very similar of what was required by the classifier. The only issue which was thought about that stage was the conversion from the COCO format exported, to the required format as an input for the classifier, had to be done manually which might prove to be tedious.

After identifying what to include in the dataset, it was time to start focusing on what images are to be collected. To have a better idea of the number of images required as part of the dataset, it was decided to do a small number of test runs using different images of the six food items. The first test was made up of eighteen images which included a mix of singular and multiple food items on the same plate. All the images throughout the study were taken by an iPhone 11, this is because past researchers made use of lower quality images whilst making use of Mask R-CNN and achieved excellent results. Using a Python script, each image was decreased to 81% of its actual size, from 4032x3024 to 3264x2448 if landscape, and 3024x4032 to 2448x3264 if portrait. The images where decreased to 81% of their original size because in a research conducted by Pokhrel Binayak as part of his internship with Leapfrog Technology, a similar approach to this research was taken and the researcher achieved promising results whilst using the previously mentioned image dimensions. These where then annotated using the previously mentioned tool and prepared for the classifier. Since Instance Segmentation was required, it was opted to draw polygons around the food objects rather than bounding boxes. A model was then trained with one epoch for the small subset of train images whilst also making use of pre trained weights from COCO as a starting point. It was opted to use these weights from previous research as it might reduce the number of iterations/images needed due to the similarities which might be present in the previously trained weights. The trained model was then tested on three of the eighteen images which where not included as part of the training set. These images included all the food items on one plate to see if anything would be detected, but this was to no avail as nothing was detected. The second attempt made use of the model which was created for the second-year research previously mentioned about “Pastizzi”, to see if the model created previously would be able to at least detect the “Pastizzi” found in the test images and thus serve as a good starting point to continue training for the new items added to the list of foods. This test showed good results in detecting each instance with pixel-by-pixel recognition but deemed to be over-trained because all the food items in the image were labelled as “Pastizzi”, even those with little to no similarities. This most probably meant that the model trained in the previous research was over trained with one class thus providing in-accurate results.

With both previous tests showing problems with a small dataset, it was decided that a larger dataset would be needed. This opened a new problem of how these images were to be collected, as it would take a long time to do so manually. At this point the process of augmenting images was researched to see if it would be possible to perform certain augmentations on each image to create multiple variations of each without making the photo look too unrealistic. Whilst thinking about these augmentations one of the issues which was thought about was that the rotation of each object changes frequently from one photo to another in real life and that might cause issues for the machine to identify. That is why with the gained knowledge about augmentations and the need of a larger dataset, the idea to augment each image with a rotational degree of 15 degrees came to mind. Therefore, another Python script was written which accepts a degree value, the source path of the images to augment and the destination path where the images should be saved. This script then rotates the source images by the rotational degree provided as a parameter until it is rotated by 345 degrees, each image is then saved to the destination folder also provided as a parameter. So, at that point, each image which was manually taken was turned into twenty-four images, making it easy to create a large dataset utilising the same photos, but this unfortunately provided further issues. Now that a large dataset could be created, this meant that the problem of manually annotating each image and adjusting the COCO annotations would become an extremely tedious task to go through. This is when a solution was formed to not only rotate each individual image by a degree, but also rotate the annotated polygons. With the use of sine, cosine, and tangent one would be able to find the new location of each point of the polygon from the origin of the image. The new annotation file would then saved with a postfix of the rotational degree they would have been rotated, for example if the annotation file was called “annotations.json” the new file would be called “annotations\_15.json” depending on the rotational value. To confirm that this script was working as intended, another small script which drew the newly rotated polygon on the rotated images was also created, this gave assurance that whatever image provided, landscape or portrait, the newly rotated annotations would be mapped correctly. If this where to be incorrect, it would cause greater problems when training the model as it might be training on polygons which are in the wrong position. The final step in this Python script, is to convert the format into the required COCO format. During this stage one would have been left with numerous rotated annotations files and their respective rotated images, but the classifier only accepts one JSON file not numerous ones, therefore a simple merge script was created which can be executed on a particular directory where all the annotation files could be found and combines them all into one file, leaving only the augmented images and all their annotations in one file. It is also important to note that the initial set of manual photos still had to be taken and annotated, so even though this automated most of the process, there was still some manual work to be done, but it took little time with the help of the created scripts.

After running the augmentation scripts, a larger dataset was created, and it consisted of four-hundred and thirty-two images. These images where all to be used for training except for the previously used three images (which now have turned into 72), to see if the results would improve when compared to first test done by training on the original fifteen images only. A model was then trained, using the pre-trained weights as stated previously and using the newly increased dataset for one epoch the same as before. This produced better results because the “Qassatat” class was identified correctly in the same image which previously resulted in zero detections, it was then opted to re-create the same exact test, but with the difference that the config would be change to four epochs. Since the training was being done on a *Core i5, 8GB RAM* Laptop with no independent graphics card, the training for one epoch was taking anywhere between two to three hours, so the number of epochs couldn’t be increased to a large number such as fifty. Therefore, it was opted to increase the iterations to four instead of one, and then analyse if there would be any decent progress. Previous steps were re-executed to train a new model with the new configurations. With all the changes done, the final testing model with four epochs showed extremely promising results where all the instances where individually detected but one of the classes was not detected correctly. This was not a problem because at that stage all that was important to understand is what size of dataset would be considered too small or too large to return decent results. From the tests which where conducted, it was finalised that the structure of the data set and the scripts created could be implemented and the actual data collection could start as all the information gathered was enough to understand fully what the final dataset would require.

Since eighteen images produced a total of four-hundred and thirty-two images, it was opted to manually take around sixty images. Previously in the testing phase, since only the concept was of importance and not the actual results, no fake objects were included in the dataset, but when the actual set was being these where also included. These fake objects were made up of common household foods such as slices of bread, green apples, grapes, and kiwis. By adding these items, it made the dataset to have more realistic scenarios whilst also giving the opportunity to test the trained model with other never seen before objects. With these it was made possible to see whether the final model would be capable of realising that the fake objects are not one of the six classes or if it mistakenly identifies them for one of them. A mix of photos where taken, where some images were composed of a single labelled/non-labelled food item and multiple items mixed with labelled and non-labelled items on the same plate. Once these photos were finalised, they were re-sized and annotated as previously discussed, but instead of immediately augmenting both the images and their annotation files, it was decided to create multiple configurations for the dataset where each configuration would split the images into test and train with different random seeds. This extra step was done to make sure that when the images where split into test and train using randomness, there would be less chance that by luck, the seed which was used gave the best possible result by coincidence and all of the other un-used seeds would provide far worse results. It is also important to note that even though randomness was used, providing seeds was of utmost importance, so that if one would like to re-produce the same executions and results, that would also be possible. A total of three configurations were created and in each of the three configuration directories, the dataset was split into a ratio of 70% train and 30% test as it is one of the most common splits in a lot of research related to machine learning. At that point the dataset was finalised, and the training of the models could be done on this.

As previously mentioned, each iteration took anywhere between two to three hours, this meant that if three different models for each of the random configurations had to be trained, the number of epochs had to once again be a small number. Based on the results which were achieved from the test images in the last few attempts, it was decided to go with four epochs per configuration. The overall time taken with a learning rate of 0.001 and making use of all the layers for the classifier was that of around 30 hours. Each model was then tested using visualization of the detected masks, to get an idea of how each performed whilst using random photos from each of the respective configuration’s test directory.

Part II: Calorie Estimation

Once the dataset was created and the respective models were trained, it was time to think about the approach to be taken to estimate the nutritional values. The first thing which had to be gathered was, the nutritional values for each of the six objects per one-hundred grams. These were generously provided to me by Andrea Bartolo from the company *Threesixo*[[2]](#footnote-2), who had the dataset for these items as he offers HAACP Consultancy. He also mentioned that, since the values are not laboratory tested there can be a discrepancy of around plus or minus five percent, so if one of the objects was weighed and calculated to have 100 calories for example, the actual value can vary between 95 and 105 calories. The only values which were not provided by him where the ones for the “Ġbejniet” because *Benna*[[3]](#footnote-3), which is a Maltese company which produces fresh dairy products, had the nutritional information readily available on their website and thus it was opted to use their “Ġbejniet” and their values as part of the dataset.

Each of the objects which were photographed to be part of the dataset were also weighed in the process, to use this real weight as the ground truth value and to be used in the calorie estimation formulas. To calculate the number nutrients found in each detected item an approach of simple proportion was taken. For this part, three main ratios had to be found:

1. The total number of pixels per 1cm2
2. The total number pixels for each food item: Real value of nutritional values for the same food item
3. The total amount of nutritional values per 1cm2

For this research, it was opted to focus only on estimating the calorie values for each item detected, but the same process would be applied to any of the other nutritional values such as carbs or protein. To find the proportions of the first point, a coin from one of the images was used. This was chosen because it is a regular circle, thus the equation of was applied to find the total area. At that point, the total pixels from the mask and total centimetres squared were found, therefore simple proportion was applied to find how much pixels are equivalent to 1cm2. For the second point, using the previously found ratio, the area in cm2 for each item was found by simple proportion from the total number of pixels. Finally for the third ratio, the average weight of all the items where firstly calculated. This was then used to calculate how much calories are found in the food item on average, using simple proportion from the nutritional values per 100g’s. Once the pixel equivalency of 1cm2 and food item’s calorie equivalency in pixels was found, simple proportion was once again used to find how much calories are equal to 1cm2. The second and third point were repeated for all the 6 different food items, the first one did not have to change because the coin stayed constant all throughout the dataset. If calories were to be replaced with any other nutritional value, the only thing which would have been changed was the calculation of nutritional value per 100g’s. The figures below are an example of how the above was done for a particular food item, in this case, for “Zalzett Malti”.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Item | Weight (Average) | Calories (100g) | Calories (Average) | Cm2 (Average) | Pixels  (Average) | Pixels  Per 1cm2 | Calories  Per 1cm2 |
| Coin | N/A | N/A | N/A | 5.21 | 17441 | 3349.092px | N/A |
| Zalzett  Malti | 115g | 209Kcal | 240.35Kcal | 35.88 | 120167px | 3349.092px | 6.70 |
| Imqaret | 35g | 302Kcal | 105.7Kcal | 29.62 | 99212px | 3349.092px | 3.66 |
| Gbejniet | 90g | 289Kcal | 260.1Kcal | 30.50 | 102163px | 3349.092px | 8.53 |
| Ghaq Tal-Ghasel | 160g | 293Kcal | 468.8Kcal | 178.22 | 596868px | 3349.092px | 2.63 |
| Pastizzi | 85g | 278Kcal | 236.3Kcal | 52.71 | 176546px | 3349.092px | 4.48 |
| Qassatat | 206g | 269Kcal | 554.1Kcal | 97.32 | 325956px | 3349.092px | 5.69 |

With the calorie/cm2 and cm2/pixels calculated for each of the six food items, the next part of calculating the total estimated calorie value was to use the models created to detect food items from the testing images, extract the detected mask total pixel values and use the acquired values to calculate the calories. The main reason why the proposed method for nutritional value estimation was chosen because, the value of calories per cm2 would stay constant even if the number of pixels detected by the classifier vary are different. Having said this, the accuracy of the calories will mainly depend on how well the image instance segmentation performs as the pixels from the detected mask were used as previously described.

1. https://www.robots.ox.ac.uk/~vgg/software/via/ [↑](#footnote-ref-1)
2. https://www.threesixo.com.mt [↑](#footnote-ref-2)
3. https://www.benna.com.mt [↑](#footnote-ref-3)