UTRECHT UNIVERSITY

MASTER THESIS

FoodWasteAI: A Multi-Task Transformer Framework For Food Waste Image Processing

Author:
Abdalla MOUSTAFA

Supervisor:
Prof. dr. Albert Ali SALAH

Orbisk Supervisor:
Pieter Marsman

Second Reader:
Dr. Ronald POPPE

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Information and Computing Sciences
Faculty of Science

August 11, 2023

Declaration of Authorship

I, Abdalla MOUSTAFA, declare that this thesis titled, “FoodWasteAI: A Multi-Task Transformer Framework For Food Waste Image Processing” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:
Abdalla Moustafa

Date:
August 18, 2023

Abstract

Food waste monitoring is an important step towards reducing the amount of food lost every year. Automated solution to monitor food waste in catering institutions have shown great results in reducing food waste in their kitchen. However, developing such solutions using machine learning is difficult due to the lack of publicly-available food waste data. Through artificial intelligence, Orbisk provides data-driven insight to catering institutions on how to effectively reduce their food waste. Their hardware solution is placed at many hospitality and catering centres to collect images of food before it is thrown away. These images are passed through Orbisk's annotation pipeline where they are annotated, either by neural network models or manual annotators. As a result, Orbisk compiled a huge dataset of food waste images that is used to train their deep learning models. In this thesis, I utilise this dataset to implement a multi-task transformer framework (FoodWasteAI) to process food waste images. The FoodWasteAI architecture is inspired by the Mask2Former model but can handle segmentation, classification and regressions tasks concurrently. The model is trained and evaluated on Orbisk's data which is comprised of more than 900,000 manually annotated images. The model achieves good results on all tasks, most notably, the ingredient segmentation task achieves 27.6 mAP on 740 different ingredient labels. In addition, the model achieves 4% better performance in a test simulation as compared to Orbisk's current AI models. This is done using 35% less parameters and 25% faster training time. To summarize, this research forms a solid foundation in automated food waste processing using deep learning models.

Keywords— Food waste, Computer vision, Deep learning, Instance segmentation, Vision transformers, Multi-task learning

Acknowledgements

I would like to express my gratitude for my project supervisor Prof. dr. Albert Ali SALAH, who gave me invaluable guidance and feedback throughout the project. I am equally grateful for my daily supervisor, Pieter Marsman who was always available for discussion and advice on my progress. Special thanks to my colleagues at Orbisk Gregorio Bertozzini and Wouter Hoffman for their practical assistance throughout my work at Orbisk. I also want to thank Johanna Schacht for giving me the opportunity to be part of the ambitious and impactful project. A big thanks to everyone at Orbisk for their warm welcome and friendly atmosphere. Last but not least, I want to thank my family for their undying support and understanding. Thank you everyone.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

According to the United Nations (UN), ending world hunger is listed as the second most important sustainable goal to ensure human well-fare and prosperity [1]. However, one-third of all the food produced globally for human consumption is wasted every year, which accounts for over 1.3 billion tons of food being wasted, or 185 kg per person per year [2]. Food waste refers to food that is fit for humans to consume but that ends up being discarded [3]. Around 17% of all food produced is thrown away by consumers which contributes to one trillion euros of economic loss [2], [4]. More importantly, the Food and Agriculture Organization of the United Nations (FAO) indicate that 828 million people worldwide are facing hunger and more than 3.1 billion people do not have access to healthy diet [5]. They estimate that reducing food waste to zero would provide enough food to feed over two billion people [6].

In addition to these issues, food waste is responsible for 8 - 10% of all greenhouse gases (GHGs) that affect global warming [7]. According to the UN, if food waste was a country, it would be the third most GHGs emitter behind China and the US. Moreover, 30% of world's agricultural land is used to grow food that is never consumed [4]. Consequently, agriculture is the leading cause of deforestation. According to the World Wild Life (WWF) organization, 17% of the Amazon rain forest has been converted for cattle ranching [8]. All in all, food waste contributes to global warming and deforestation which leads to negative effects on both the economical and humanitarian levels.

In Europe, the Netherlands is the 5th worst country when it comes to throwing away food with over 161 kg per capita as compared to the European average of 127 kg per capita [9]. In 2013, the food catering industry in the

Netherlands was responsible for 446,000 tons of food waste [10]. According to Rabobank in 2020, the hospitality sector is responsible for 14% of the food waste generation in the Netherlands [11]. One of the reasons behind wasting food in that sector is the lack of information regarding what food is wasted and at what stage. In fact, food waste monitoring is one of the prominent solutions to gain insight and reduce food waste for businesses [6], [12]. In a study by Leverenz et al. (2020), the authors reported a reduction of over 64% in food wastage mass by simply implementing a self-reporting system during breakfast buffets in four German hotels [13]. Their study shows that monitoring systems can provide valuable insight and incentive for the staff to reduce food waste in kitchens.

As a result, many start-ups propose automated solutions to monitor food waste in kitchens. Orbisk is a start-up on a mission to reshape the world's food system by cutting down on food waste. The company provides valuable insight in the kitchen on what and how much food is thrown away during the different processing stages. Using an automated food waste monitoring system, Orbisk is able to empower customers with data-driven decisions on the best methods to reduce their food waste. Their AI solution consists of a camera placed on top of the waste bin that records the food being thrown away, then it sends the images to their image recognition pipeline in the cloud to perform the necessary predictions. Their clients have access to a dashboard that showcases important statistics about their food waste performance, in addition to providing action-oriented feedback on how best to lower their waste.

1.2 Orbisk AI Pipeline

Orbisk's main hardware solution is called *Orbi*, which is a camera on top of a waste-bin powered by a Raspberry Pi and connected to a scale as shown in *Figure 1.1*. The device uses a motion sensor to activate the camera whenever an object is held in front of it. Whenever the camera captures an image, the device records the weight from the scale and sends the image and weight information to Orbisk's annotation process where the deep learning image recognition models start.

Currently at Orbisk, there are two deep learning models working together to annotate the food waste images. The first model is a multi-task model that works on the whole image to predict five labels: has-waste, waste-stream,

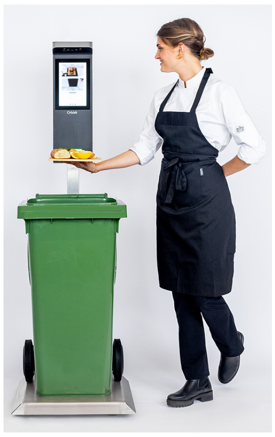


FIGURE 1.1: The Orbi with a waste bin on it. Image from [14].

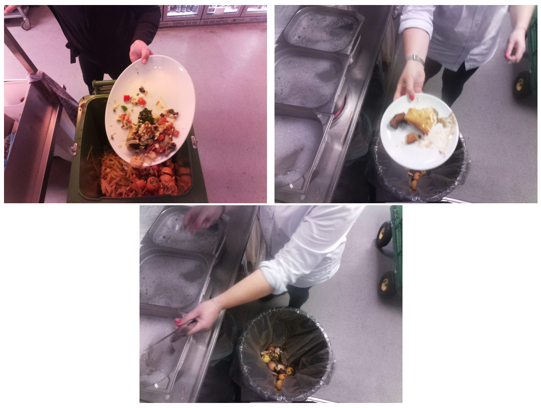


FIGURE 1.2: Sample images taken with Orbi. Image from [14].

container-type, image-quality and visual-weight. The other model is an ingredient recognition model that generates a mask and a label for each ingredient of food in the image. It is worth noting that these models run sequentially on each image and a decision rule is used to decide whether to accept their predictions or to send the image for manual annotation instead.

1.2.1 Multi-Task Model

The multi-task model at Orbisk uses a convolutional neural network (CNN) architecture called ResNeSt50 [15] as the backbone to encode the image and five head layers to perform each task. The first task is a binary classification for the presence of waste on the image. The second task is a 27-label classification problem to predict the type of container in the image. The third task is to predict the waste-stream of the food which is the stage at which the food is thrown away, whether it is served, unused, peeled, etc. The fourth task is to predict the image-quality out of six different categories. Finally, a visual weight value is assigned to the image to indicate the weight of waste present on the image.

1.2.2 Ingredient Recognition Model

The ingredient recognition model uses Mask-RCNN architecture [16] to produce a mask and a label for each ingredient instance in the image. The model works by encoding the image, then generating bounding box anchors across the image. A region-proposal network then ranks the bounding boxes according to how likely they are to contain an object and applies a non-maximum suppression algorithm to obtain the good proposals. The network then uses the remaining proposals and the encoded features to select the most appropriate bounding box and label for each ingredient in the image. A mask head is used to generate a mask for each bounding box. After that, the masks are processed to merge adjacent masks that contain the same ingredient.

1.3 Research Motivation

The current setup at Orbisk introduces a few issues in terms of performance, costs and maintenance. Addressing these issues is the focus of this project.

1.3.1 Performance

Due to the similarity of tasks that the models perform at Orbisk, there is great inter-dependency between them. Combining all tasks in one model forces the model to focus on features that are more general to all tasks instead of over-fitting on the noise [17]. Moreover, this helps the model to generalize well to new tasks in the future, since representations that perform well on many tasks, are more likely to perform well on new tasks in the same domain [18]. An earlier research I conducted at Orbisk that investigated the effect of combining three classification tasks into one multi-task CNN model showcased that the multi-task model's performance at each task is equal or better than a single-task model performing that task. The preliminary results of that study can be found in Appendix D.

At Orbisk, they use a CNN architecture as the backbone for both the multi-task and ingredient recognition models. While CNNs established themselves in many vision tasks, Min et al. (2020) concluded that the discriminative details in food images are too subtle for current CNN-based architectures to distinguish [19]. Indeed, classifying food ingredients is a highly challenging task due to the intrinsic variability in food appearance [20], [21]. Different food ingredients, e.g. sugar and salt, can appear very similar in images [22]. In addition, most food ingredients are non-rigid, so they can easily deform to various shapes.

In computer vision, local features of an image correspond to patches of nearby pixels, while global features refer to the image as a whole [23]. While local features are useful to many computer vision tasks, classifying food images can benefit from modeling global features as they allow the model to distinguish the subtle differences between food instances [19]. While CNNs are perfectly capable of recognising the local features of an image, they do not perform well when trying to detect global features [24].

On the other hand, transformers are inherently able to model global features due to their ability to process the input as a whole using positional encoding and self-attention [25]. In NLP tasks, they showcased a much higher performance as compared to other neural network architectures such as long-short term memory (LSTM) and recurrent neural networks (RNN). Meanwhile in computer vision tasks, vision transformers (ViT) outperform CNNs in various high-level problems such as classification and segmentation [26]. More recently, newer models adapted new methods to incorporate global self-attention with local ones in order to further enhance the performance [27]–[29]. This thesis aims to contribute to these efforts by investigating ViT-based networks in food waste image processing.

1.3.2 Costs

With the current design at Orbisk, the two models need to be re-trained on the new data every few months. This is to ensure that the models incorporate data that come from new customers. Moreover, two forward passes per image are needed to obtain the required labels. Having one model that predicts all the needed labels will save training time. Depending on the design, training time may reduce by up to N, where N is the number of tasks combined. In addition, inference costs on cloud and price per image will be reduced since each image passes through one model instead of two. All in all, a multi-task model will greatly save deployment costs and makes it faster to re-train and replace running models.

1.3.3 Maintenance

Currently, the code base for the models are separate. In addition, it contains many duplicate code (e.g. data-loader class) to allow the training and inference for each model. Implementing a multi-task model will simplify the code base and makes it easier to maintain. Loading data, creating model and training will be done once which simplifies the code logic and its readability.

Furthermore, updating the models in the pipeline would only involve training, packaging and deploying one model instead of two. All of this reduces thetime needed for maintenance.

1.4 Research Approach

1.4.1 Research Questions

This thesis focuses on three objectives. Firstly, I aim to apply single-task transformer-based models to a complex and imbalanced dataset of food waste images to examine its performance and cost. Secondly, I aim to study the effectiveness of combining the classification and regression tasks in one model and compare it to a multi-task variant that combines all tasks. Finally, I aim to improve upon Orbisk's current AI pipeline by proposing one model that handles all the tasks and trained end-to-end. Accordingly, the thesis address the following question:

"Can a multi-task transformer framework be used to jointly handle segmentation, classification and regression tasks in food waste images without sacrificing accuracy?"

Currently at Orbisk, there are four classification, one regression and one segmentation tasks that are performed by their AI pipeline. One of the primary objective of this research is to investigate the effectiveness of combining all these tasks into one model in terms of accuracy and costs. This means a baseline is needed to compare the multi-task models with. Therefore, the thesis addresses the following sub-question:

RQ1: "How do the single-task transformer models perform in terms of accuracy and cost on each task?"

At Orbisk, they only combine the classification and regression tasks together into one model, leaving the segmentation task to be done by a dedicated model. The thesis aims to understand whether this setup is ideal for their tasks. More specifically, the aim is to understand whether combining all tasks into one model improves the performance for all tasks as compared to only combining the classification and regression tasks together. Hence, the thesis addresses the following sub-question:

RQ2: "Does combining all tasks in one model yield better performance than separating them into two models?"

Finally, a comparison between the proposed multi-task framework and Orbisk's current setup is conducted using a test simulation. The simulation analyses how the models perform in a production environment using unseen data. Hence, the following sub-question is addressed in this thesis:

RQ3: "How does the proposed multi-task framework which combines all tasks perform in a test simulation with real-world data?"

1.4.2 Research Method

RQ1 concerns the application of single-task models that is trained for each task at Orbisk. These models will be used as a baseline for all the subsequent experiments. RQ2 aims to provide a deeper understanding on how these tasks relate to each other. Two model instances will be trained. The first one combines all classification and regression tasks and trains them together in one model. The second model will be trained on all tasks. These models will be compared to each other and to the models proposed in RQ1. Finally, the best multi-task framework will be compared to the current setup at Orbisk to assess how it performs in the real world setup. A simulation experiment using unseen data will be conducted to both the proposed multi-task framework and the current setup to determine which one performs best.

For both RQ1 and RQ2, the models' performance will be validated using the accuracy metric for all classification tasks, weighted mean absolute percentage error (WMAPE) for the regression task and mean average precision (mAP) metric for the segmentation task. For RQ3, the picture accuracy and food picture accuracy of the models will be used. These metrics are defined in Section 3.3.

1.5 Thesis Outline

The thesis is organised as follows. Firstly, previous work relating to food waste monitoring, image recognition and multi-task learning is discussed. Then, the methodology section describes the datasets and models to be used along with their detailed architecture. Thereafter, the experimental setup along with the ablation studies are outlined. Afterwards, the results of these experiments are presented. Finally, a discussion of the results is conducted along with the limitations and future work, before ending with the conclusion.

Chapter 2

Related Work

2.1 Food Waste Monitoring

Food waste monitoring and tracking provides necessary information for detailed prevention plans and helps in tracking the performance of such plans [30], [31]. There has been a lot of research in this field to measure how effective food waste tracking is for addressing the food waste issue. Bharucha (2018) conducted a qualitative study with 63 restaurants in Mumbai to investigate the problem of food waste in restaurants and how they are handling it [32]. He interviewed restaurant owners to enquire about how the food waste is handled at their property. His study showed that 75% of restaurants prepare 10 - 20% more food than they actually need. High-end restaurants prepare even more food in case they need to serve additional crowds than previously estimated. Therefore, these restaurants are more incentivized to participate in food waste reduction initiatives. He concluded that micro-management solutions are more effective than large-scale ones in reducing food waste in Indian kitchens.

Another study conducted by Filimonau et al. (2020) explored the food waste management practices in 22 Shanghai restaurants [33]. The study outlined that preparation is the most wasteful phase of restaurant business. Moreover, all restaurants were unable to provide exact figures on the amount of food wasted at their venues. The authors highlight that the lack of data is a big obstacle towards effective mitigation policies. Eriksson et al. (2019) studied the effect of food waste quantification on food waste reduction [34]. The authors compiled data from 735 restaurants, canteens and hotels in Europe that use different techniques to record their food waste. Their methods of tracking ranged from spreadsheets to internet-powered services. The results of the study show that 61% of restaurants were effective at reducing their food waste output with the help of their food waste quantification set-ups.

2.1.1 Qualitative Food Waste Monitoring

Many contributions to monitor the amount of food waste at various levels of the supply chain collect data through participant interviews or self-reporting from the target venues [13], [33], [35], [36]. A study by Filimonau and Ermolaev (2021), showed that the restaurant sector in Russia contributes around 7% of of the country's total food waste output [37]. The study recruited 21 dining facilities across different food service categories in Kemerovo, Russia. During the study, the authors interviewed the top executives of these facilities to give feedback on the state of food waste at their venues. The authors discovered that the 57% of the interviewees mention plate waste as a main driver for food waste, while 52% of them mention over-production of meals as a main driver. In addition, to set a benchmark for the food waste generated in various categories of the Russian food services, the authors used the financial records of food waste collection provided by the study participants. In Hungary, Filimonau and Sulyok (2021) conducted a similar study where they asked participants to evaluate the food waste problem at their venues [38]. Their results were consistent with prior studies in identifying plate waste and overcooking as the main drivers of food waste.

2.1.2 Fully-Automated Food Waste Monitoring

Another method to monitor food waste is through automated devices that track food as it is thrown. Martin-Rios et al. (2021) conducted a study to understand the use of technological advancements to monitor food waste in hospitality, restaurant and catering services [39]. They highlight that most of the monitoring systems in the mentioned industries use manual labor to identify the amount of food waste generated. This is done by three main mechanisms: (1) having a staff member stand next to the trash bin to record the food waste, (2) having different colored trash bins to categorise the food waste then weighing them at the end of the day, and (3) having a tablet attached to a scale where the staff can place the food and type in the correct category of waste before disposing it. In addition, the authors discussed five fully-automated solutions to track food waste, namely LeanPath, LightBlue, Winnow Solution, Kitro, and Orbisk. They concluded that the use of technology is particularly important for the food service industry as it provides executives with insight on how to optimise their catering processes. Their conclusion is further supported by the study done by Eriksson et al. (2019) where they indicated that catering services that use more automated set-ups to measure their food waste recorded more data and were able to reduce their food waste more [34].

2.2 Deep Learning in Computer Vision

Deep learning is a subset of machine learning in which models are designed to automatically select useful features to make good predictions. Classical machine learning algorithms rely on structured features that are usually defined by experts to make predictions. Deep learning algorithms, however, can automatically "learn" from the data which features are useful to make correct predictions for their task. This is done using multiple layers that provide different level of abstraction for the data [40]. As a result, deep learning models have dramatically improved the state-of-the-art (SOTA) performance on many tasks, such as speech, visual recognition, and many other domains such as genomics. In this section, I outline the relevant deep learning approaches mentioned in this thesis. In addition, I provide an overview on how they are trained and optimised on data.

2.2.1 Artificial Neural Networks

Artificial neural networks (ANN) are the heart of deep learning algorithm. These neural networks are modeled after the brain cells using an artificial neuron called the perceptron. The perceptron was first proposed by Rosenblatt (1958), where he outlined the basic principles of its design. A perceptron is modeled after the brain cells and is able to capture the statistical measurements obtained from a large volume of data. Neural networks make use of it by defining layers with many perceptrons where each one in a layer is connected to all perceptrons in the previous and following layers as shown in *Figure* 2.2. The first layer is called the input layer, which takes the inserted data. The final layer is called the output layer and it is responsible for making predictions. The layers in-between the input and output layers are called hidden layers.

The ANN is called deep due to the presence of an arbitrary number of hidden layers that provide multiple level of representations for the inserted data. The connections between the perceptrons from one layer to the other are

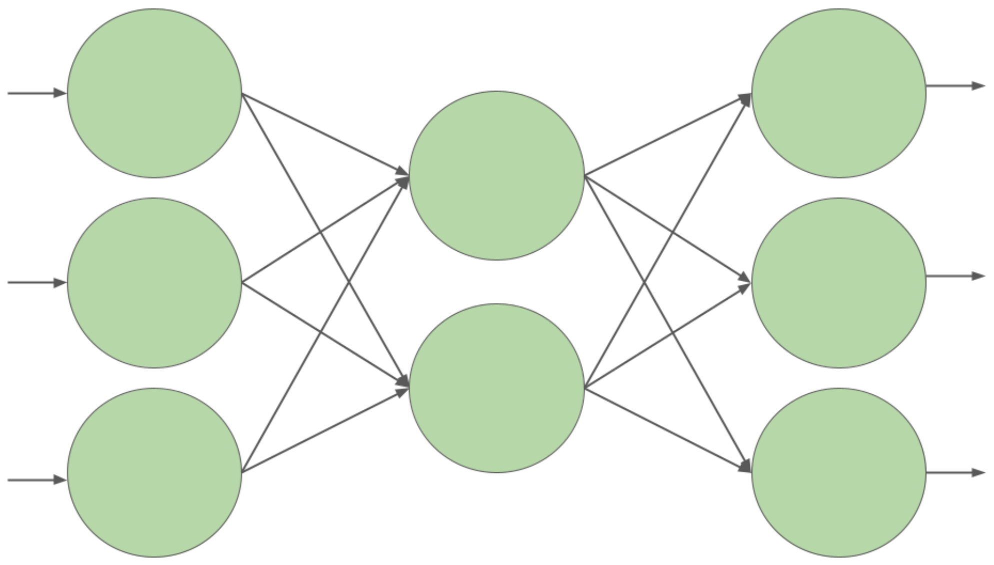


FIGURE 2.1: Artificial neural network with input layer, one hidden layer and one output layer.

called weights. Similar to brain cells, these weights are responsible for understanding the relationship between the features and are, therefore, learned using a large volume of data. Each perceptron accepts an input and process it to provide an output to the following layer. The processing of each perceptron is governed by an activation function that decides the value that is passed through to the next layer. There are many types of activation functions that are used depending on the task, such as ReLU [41], Sigmoid and Softmax [42]. Their exact formula can be found in *Appendix A*.

2.2.2 Convolutional Neural Networks

Convolutional neural networks (CNN) are a type of ANNs that specialize in data with a grid-like structure, such as images. The CNNs replace the conventional perceptrons with kernels that are applied on the input. These kernels are matrix-like filters with learnable weights that are adjusted during training to fit the processed data. A CNN contains at least one convolutional layer which consists of an arbitrary number of fixed-sized kernels. Given an input of X with size $\mathbb{R}^{nxn}$ and a kernel K with size $\mathbb{R}^{mxm}$, the output is a feature map F of with size $\mathbb{R}^{qxq}$, where $q = \frac{m-n+2p}{s}$, p is any padding applied to the feature map and s is the stride that the kernel takes on the input X. Each value in F is the summation of element-wise multiplication between the kernel and input. More formally, $F_{x,y} = \sum X_{i,j} \odot K$ where $F_{x,y}$ is the element in F at position x, y, $X_{i,j}$ is a sub-section of X that overlap with K and $\odot$ is the

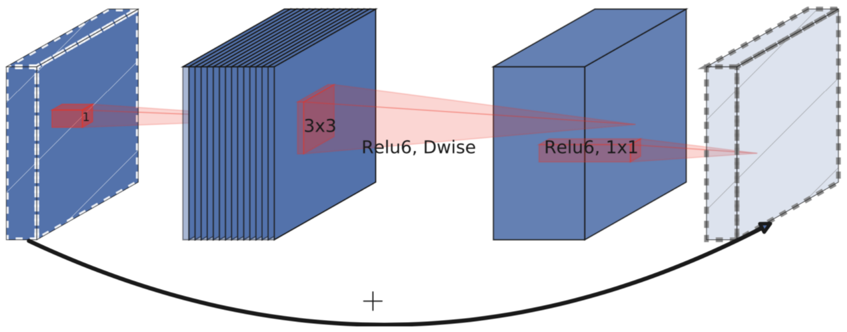


FIGURE 2.2: CNN architecture. Image from [43].

element-wise operation.

2.2.3 Vision Transformers

Vision transformers (ViT) are another type of ANNs proposed by Dosovitskiy et al. (2021) [44] where they successfully trained a transformer model on images and attained strong results. Transformer models were originally developed for natural-language processing (NLP) tasks in which the text is fed to the model in a sequence of tokens. A key characteristic of transformer models is the use of encoders and decoders. Encoders are blocks of consecutive layers that aim to provide a good representation of the input. The decoders take that representation and process it to obtain the desired output. Each encoder or decoder consists of mainly two layers, a self-attention layer (SA) followed by a multi-layer perceptron (MLP). There might be additional layers depending on the architecture.

The self-attention layer is a mechanism that enables the model to obtain the global dependencies between data. It allows the model to focus on different regions of the input depending on its relevance to the task at hand. ViTs have two additional component to help represent an image, namely the patch and positional embeddings. Each image is split into smaller patches and each patch is represented by an embedding vector. Each embedding vector is combined with the positional embedding of the corresponding patch before being fed to the model.

2.2.4 Training

All the ANN-based methods have weights between their layers which determines how well the models' predictions are. Optimising these weights is

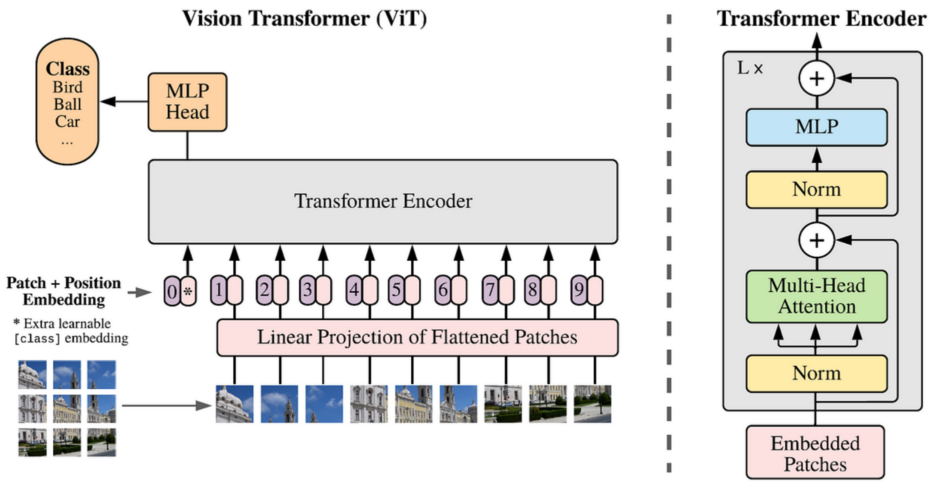


FIGURE 2.3: ViT architecture. Image from [44].

the training phase of the model. The weights are optimised using an algorithm called back-propagation proposed by Rumelhart et al. (1986) [45]. This procedure iteratively adjusts the model's weight using a quantifiable error obtained from the model's predictions and the ground-truth labels. The error can be obtained using a variety of algorithms depending on the task. For instance, classification tasks tend to use categorical cross-entropy loss (*Equation A.1*.

After obtaining the loss, the back-propagation algorithm is used to find the weights' gradients by moving backwards through the model. Partial differentiation is used to arrive at the gradient value of each weight that minimizes the loss. After finding the gradient of each weight, a gradient descent algorithm is used to update the model's weights. The gradient descent algorithm was first outlined by Cauchy et al. in 1847 [46]. Since then, a variety of similar algorithms are developed, such as Adam [47] and Adagrad [48]. The standard gradient descent algorithm is shown in *Equation 2.1*.

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} L(\boldsymbol{w}) \tag{2.1}$$

where the w_i is a weight value in the model, $L(\boldsymbol{w})$ is the loss function used to train the model and α is the learning rate of the model which is a hyperparameter.

To summarize, the data is fed to the model in batches and the model analysis the patterns in the data and makes predictions for every data sample. These predictions are then compared to the ground-truth label to obtain the loss value. The back-propagation algorithm uses this value to calculate the weights' gradients that minimize this loss. A gradient descent algorithm is then used to update the model's weights in order to produce better predictions.

2.2.5 Transfer Learning

Generally, the deep learning algorithm is trained in a supervised manner on a specific dataset that contains data for a specific task. For instance, a CNN model developed to recognize cats and dogs would be trained on a dataset with cats and dogs images. The dataset is usually split into training, validation and test sets. The training set is used to train different model versions. The validation dataset comprises unseen data that is used to evaluate each version to identify the best performing model. Finally, the test set is used to get a general understanding of the model's performance on unseen data. However, with certain tasks, the amount of available data can be very small. This means that splitting the dataset into three subsets would leave very few images for training.

One solution to overcome this issue is using transfer learning [49]. Transfer learning is the process in which a model trained on a similar but non-identical task is fine-tuned on another task using a fewer number of data. For instance, a model trained to recognize cats and dogs with great accuracy can be fine-tuned to classify different car types. The fine-tuning process involves loading a model with it pre-trained weights then training it on a new dataset with fewer samples. This way, the model's weight are not randomly initialized and instead are trained to handle a task domain. The fine-tuning process then slightly adjusts the weights to better fit the type of task within this domain. Transfer learning has been applied in various applications to utilize the strong performance of foundational models on many tasks [50], [51].

2.3 Food Recognition in Deep Learning

Fully-automated food waste monitoring solutions require food (waste) images. Due to the limited work on food waste recognition using deep learning, this section focuses on food recognition. The similarity between food and food waste images offer comparable challenges and insight on how to address them. The introduction of AlexNet [52] in 2012 presented a breakthrough in the field of computer vision and its applications. This opened the door for many contributions to address a variety of topics. Kagaya et al. (2014) conducted one of the earlier studies on how deep learning can be effectively applied to food recognition [53]. Since then, more research has been made to improve upon their work, either by collecting diverse data or utilising state-of-the-art architectures.

2.3.1 Food Recognition with Convolutional Neural Network

Food recognition using CNNs research has seen many notable contributions over the past years. Kagaya et al. (2014) applied a CNN-based model to a custom-build dataset with 10 food classes [53]. Using local response normalisation, they managed to obtain a score of 73.7% accuracy. NutriNet is proposed by Mezgec and Seljak (2017), where they modified AlexNet and applied their model on various food image datasets [54]. Their results show that the model was able to classify 520 different food items with an accuracy of 86.72%, while using fewer parameters than AlexNet. Another work by Pouladzadeh et al. (2017) proposed a mobile food recognition system that classifies 30 different categories of food [55]. The system is able to detect multiple food instances in the same image using region proposal algorithms to generate candidate regions before ranking them based on the extracted CNN features and predicted food classes. Their system achieves an accuracy of 94.11%. Similarly, FoodAI was proposed by Sohoo et al. (2019) as an API service to power a component of a mobile App developed by the Singapore's Heath Promotion Board [56]. Their food recognition system uses an ensemble of SENet [57] and ResNeXt-50 [58] to classify 756 different food items with an accuracy of 83.2% top-1 accuracy and 95.7% top-5 accuracy. The model was pre-trained on ImageNet and fine-tuned on 400,000 food images.

2.3.2 Food Recognition with Transformers

With the introduction of the transformer architecture to the field of computer vision, more research was dedicated to investigate the architecture's capabilities in food recognition. Zhu et al. (2020) successfully used a supervised transformer network to classify food items in refrigerator images [59]. They proposed a RectNet architecture to rectify the refrigerator images before using a fully convolutional network (FCN) for recognizing each food item in the image. Their results show a 3 - 5% improvement compared to other baseline methods. Another work by Song et al. (2022) introduced a noise-robust locality transformer for food image retrieval [60]. Their approach addresses the fine-grained characteristics of food images by proposing a patch attention module (PAM) and local perception unit (LPU). PAM aims to lower the impact of the noise by redistributing low weights to noisy patches, while LPU uses convolution to extract local features to obtain fine-grained information. The authors applied their approach on the Food-101 dataset in which their method obtained 8% higher mean average precision compared to the Swin-small [61] architecture.

In another work, LTBDNN was introduced by Sheng et al. (2022) as a lightweight transformer framework for food image recognition [62]. The authors proposed a token generation method that uses the convolution operation on the original image then applies the unfold operation on the corresponding feature map. As a result, the model is able to learn global representations and spatial-inductive bias. Moreover, the authors used a transformer grouping technique which utilises two transformers in parallel with bridge connection. This method allows the model to extract food image features from different perspectives simultaneously, which helps it learn global features of food images more comprehensively. Wu et al. (2021) introduced a multi-model framework for food segmentation using a recipe learning module (ReLeM) to incorporate recipe information [63]. The authors compiled food images from Recipe1M to create the FoodSeg103 dataset and tested various CNN and Transformer-based methods on it. Their results show that applying the ReLeM module improves all the models' performance.

2.3.3 Food Datasets

Over the years, the scale of food datasets has grown quickly. One of the earliest compilation of food images was done by Matsuda et al. (2012) in their UEC Food100 dataset, which contains 12,740 images in 100 dish categories [64]. Bossard et al. (2014) introduced the Food-101 dataset, which contains 101,000 images for 101 classes [65]. In 2020, the ISIA Food-500 was introduced which scaled-up the number of food images by a factor of four by having 400,000 images belonging to 500 categories [19]. More recently, Food2K dataset was introduced which contains over one million images belonging to 2,000 categories making it the largest food recognition dataset [66]. Moreover, there are recipe-related datasets like Recipe1M which contains 900,000 images and one million recipe [67]. Recipe1M+ is a subsequent work that expands the dataset to 13 million images. However, the last two datasets are build mainly for recipe generation research as they do not have category labels for images. In addition, the other datasets contain only image-level categories without further annotation to support ingredient-level classification and segmentation.

There are a few datasets that contain food mask annotations. UNIMIB2015 dataset was created to enable food recognition and left-over estimation research [68]. The dataset contains 1,000 images of food trays before and after consumption with 15 distinct food categories. A subsequent work by Ciocca et al. (2017) produced UNIMIB2016 dataset, which has 1,027 food tray images with 73 food classes [69]. UECFoodPix [70] and UECFoodPixComplete [71] are two datasets that contain 10,000 food images and their corresponding masks in order to facilitate segmentation research. Wu et al. (2021) introduced an ingredient-level food dataset called FoodSeg103 that contains 7,118 images with more than 40,000 masks spanning 103 different food items [63].

Although these datasets provide more fine-grained information about the contents of food images, they still lack the necessary annotations that are needed for the tasks at Orbisk. In addition, the datasets do not contain food waste images, which will hinder the model's performance. Therefore, in this thesis, I utilise Orbisk's internal dataset to train and evaluate my deep-learning framework. Orbisk has more than 900,000 images manually annotated with ingredient-level masks and labels, image-level food mask and categories for the container-type, waste-stream and image-quality. At the end of my work, I aim to disclose a subset of Orbisk's data to facilitate future research in food waste monitoring.

2.3.4 Summary and Limitations

To summarize, recent work on food recognition in deep learning focused on two main methods to improve food classification and segmentation. Many contributions such as [19], [62], [72] identified the need for incorporating global features to provide more meaningful representations to the model. These contributions showed that global features provide more useful clues to the model to distinguish between food items that look similar in local regions. Furthermore, other work focused on providing context alongside image features to help the model distinguish between food items. In a work by Bettadapura et al. (2015), the authors used the location and extra information about the restaurant as features to classify the food [73]. Their results show a huge improvement when these information are used alongside food images.

Dataset	No. of Images	No. of Classes	Label	Dish / Ingredient
Food-101	101,000	101	C.L.	Dish
ISIA Food-500	399,726	500	C.L.	Dish
Food2K	1,000,000	2,000	C.L.	Dish
Recipe1M	887,706	-	Recipe	Dish
Recipe1M+	13,735,679	-	Recipe	Dish
UEC Food100	12,740	100	B.B.	Dish
UNIMIB2015	1,000	15	B.B.	Ingredient
UNIMIB2016	1,027	73	B.B.	Ingredient
UECFoodPix	10,000	102	Mask	Ingredient
UECFoodPixComplete	10,000	102	Mask	Ingredient
FoodSeg103	7,118	103	Mask	Dish + Ingredient
Orbisk	900,000	783	Mask	Ingredient

TABLE 2.1: Summary of all food datasets. The label column indicates the type of annotation each image has. B.B. stands for bounding box and C.L. means class label. The last column indicates whether the dataset has dish or ingredient level annotation.

Paper	Model	Dataset	Task
Kagaya et al. (2014) [53]	CNN	Custom	Classification
Mezgec and Seljak (2017) [54]	CNN	UNIMIB2016	Classification
Pouladzadeh et al. (2017) [55]	CNN	Custom	Segmentation
Sohoo et al. (2019) [56]	CNN	Custom	Classification
Zhu et al. (2020) [59]	ViT + CNN	Custom	Segmentation
Song et al. (2022) [60]	ViT	Food-101	Image retrieval
Sheng et al. (2022) [62]	ViT	Food-101	Classification
Wu et al. (2021) [63]	ViT	FoodSeg103	Segmentation

TABLE 2.2: Summary of contributions on food recognition in deep learning.

In addition, the work by Wu et al. showed how recipe information provides more context for the model to perform better on food segmentation [63].

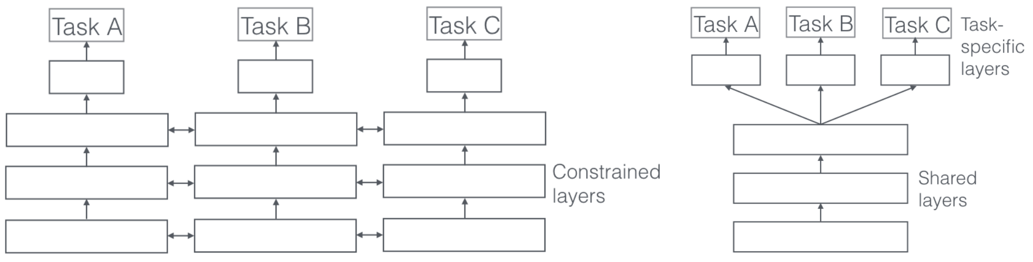
Given all this work, there are still limitations facing automatic food image processing. Most public datasets and frameworks focus on classifying food dishes in images by providing a dish label for each image. Such a label does not provide fine-grained information regarding the ingredients presented in the image. Moreover, other contributions tried to address ingredient recognition but suffered from low performance measures due to the high visual similarity between ingredients.

2.4 Multi-Task Learning with Neural Networks

In deep learning, neural networks are usually trained on one task e.g. classification and segmentation. The training is done using an optimisation function, for instance cross-entropy loss, to quantify the error of the model during training. This type of learning is called single-task learning (STL). Although this is a popular method to address important tasks using deep learning, in a setting with multiple models performing different tasks with the same data, this approach disregards any relationship between these tasks. While acceptable results can be achieved learning one task at a time, the model's performance on each task is limited due to the inability to leverage the interdependency information between the tasks [17].

On the other hand, multi-task learning (MTL) involves the optimisation of more than one criterion. This learning process aims to create models that can handle multiple tasks, while leveraging task-specific signals contained in the training data to improve model's generalization [74]. Consequently, the model is able to form a more general representation of the input data that represents all learned tasks. MTL is used across many domains, such as natural language processing [75] and computer vision [76]–[78] to improve the generalization and performance of deep learning models.

There are two main paradigms to implement MTL for deep learning, soft- and hard-parameter sharing. Soft-parameter sharing uses a model for each task, but regularises the distance between the models' parameters to encourage them to be similar using either trace norm [79] or l_2 norm [80]. Alternatively, hard-parameter sharing uses the same hidden layers for all tasks but adds task-specific output layers. Caruana (1993) was one of earliest researchers to propose this setup, where he showcased that it provides domain-specific inductive bias which contributes to more powerful learning [81]. He showed that this paradigm of MTL reduces overfitting on the shared parameters, i.e. the shared hidden layers, as they need to build good representations for all tasks. In another work, J. Baxter (2000) showed that the risk of overfitting on the shared parameters is an order N (N is the number of tasks) smaller than overfitting on the task-specific ones [18]. Moreover, sharing some hidden layers between all tasks greatly reduces the number of parameters to optimise during training.



(A) Soft-parameter sharing (B) Hard-parameter sharing

FIGURE 2.4: Simplified soft and hard parameter sharing architectures. Image from [17].

2.4.1 Learning Task Relationships

Since the introduction of the hard-parameter sharing paradigm by Caruana (1993), it has been the default choice for many MTL research. Long et al. (2015) propose an improved method to address task relationships [82]. They developed a multi-linear relationship network (MRN), which can be seen in Figure 2.5, that uses matrix priors between the task-specific fully connected layers to allow the model to learn the relationships between the tasks.

Another interesting work by Misra et al. (2016) starts out with two separate model architectures then use a cross-stitch unit that helps the model determine how to share information between the networks [83]. The cross-stitch unit uses a linear combination of the output of the networks' layers to learn the optimal combination of shared and task-specific representations. The model architecture can be seen in Figure 2.6.

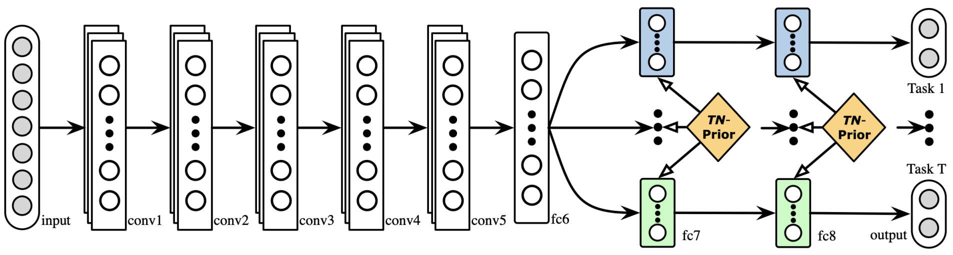


FIGURE 2.5: Multi-linear relationship network architecture. Image from [82]

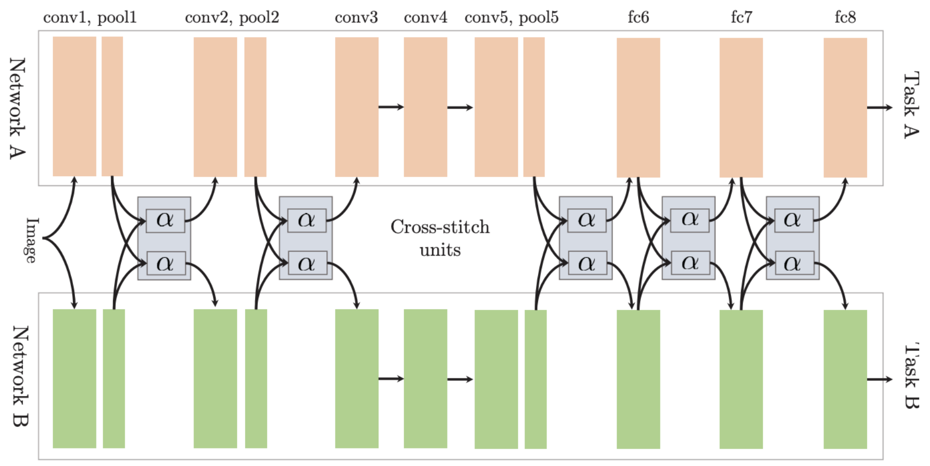


FIGURE 2.6: Cross-Stitch architecture. Image from [83]

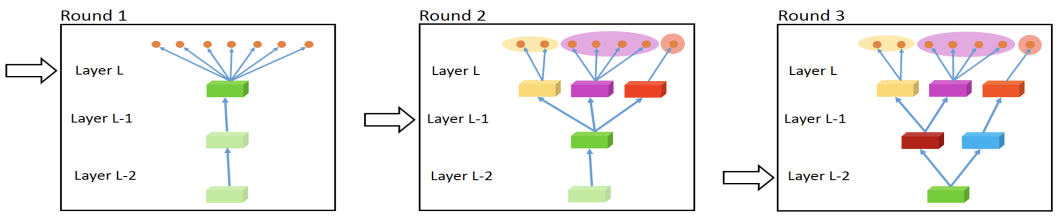


FIGURE 2.7: Fully adaptive feature sharing network. Image from [84].

However, this setup still relies on a pre-defined sharing structure which can prove limiting for novel tasks. Therefore, Lu et al. (2016) proposed a bottom-up approach that starts with a thin network structure then applies a greedy algorithm during training to incentivize the grouping of tasks as can be seen in Figure 2.7 [84]. This method is limited however, since the greedy algorithm may not yield globally optimal solution. In addition, this setup prevents the model from learning more complex relationships between tasks since it assigns a branch to each task.

2.4.2 Multi-Task Learning in Computer Vision

Deep-learning approaches revived the research in MTL in computer vision. Many contributions focus on combining different tasks, for example classification, semantic segmentation, instance segmentation and regression, in one model to improve performance and efficiency. Teichmann et al. proposed a deep learning model called MultiNet capable of handling semantic segmentation, classification and detection [85]. In another work by Sermanet et al.,

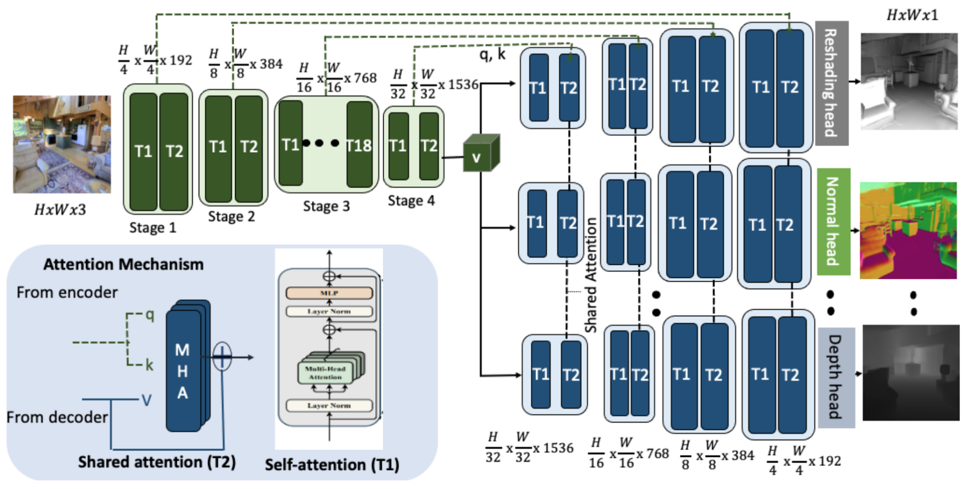


FIGURE 2.8: Multi-task transformer architecture. Image from [77]

the authors developed a CNN-inspired network for classification, localization and detection [86]. Scene understanding also saw many notable contributions such as [87] and [88]. All these approaches use a hard-parameter sharing design as basis for their models.

More recently, transformer-based frameworks for multi-task vision tasks took over CNN as the state-of-the-art models. Hu and Singh (2021) introduced UniT which is an end-to-end framework that handles many tasks across different domains [89]. The framework encodes the input using a modality-specific encoder then feeds the representations to a shared decoder before making predictions using task-specific heads. The authors evaluated the model using seven tasks and eight datasets spanning different domains like object detection, language only tasks and vision and language reasoning tasks. Their framework produced strong performances on each task with notably fewer parameters. Similarly, Bhattacharjee et al. (2022) proposed an end-to-end multi-task learning transformer that uses a shared attention mechanism between the model's decoders in order to learn the dependencies between tasks [77]. As can be seen in Figure 2.8, the model consists of a shared encoder transformer with four transformer stages inspired by the Swin architecture [61]. All task decoders have the same architecture with four transformer stages but different task-specific heads. Their results show that the transformer-based model outperforms the state-of-the-art CNN baseline [90].

Another work by Ye and Xu (2022) introduced an inverted pyramid multi-task transformer for dense scene understanding (InvPT) [78]. The authors proposed an architecture that consists of three stages, shared encoder, task-specific preliminary decoders and an InvPT decoder as seen in Figure 2.9. The encoder takes an image and outputs encoded representations that are shared between all tasks. The self-attention mechanism in transformers helps obtain global feature representations, therefore the authors experimented with ViT [44] and Swin as encoders [61]. The task-specific decoders consist of two blocks, each containing a 3 x 3 convolution layer followed by batch normalisation and ReLU activation function (Conv-BN-ReLU). These decoders produce task-specific features and preliminary predictions. These features and predictions are combined and concatenated to be fed as a sequence to the InvPT transformer decoder.

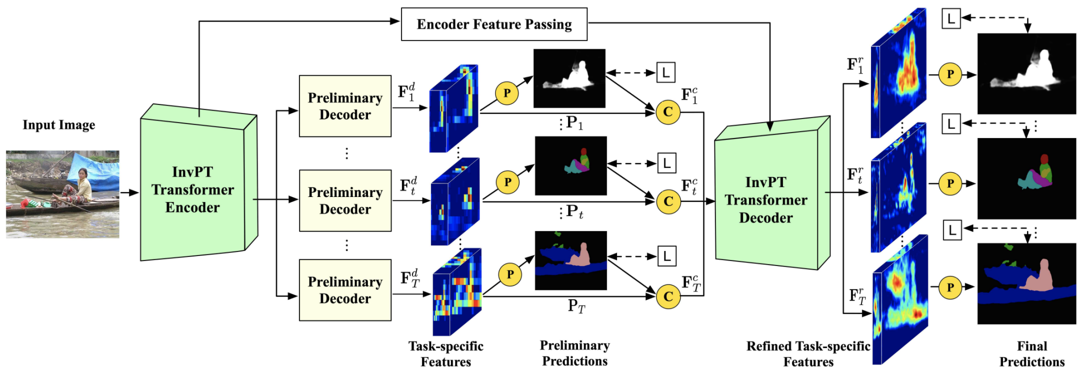


FIGURE 2.9: Inverted pyramid multi-task transformer architecture. Image from [78]

The InvPT decoder uses UP-Transformer blocks together with cross-scale self-attention message passing and multi-scale encoder feature aggregation in a unified network module. As a result, the model learns refined task-specific representations within global spatial and task contexts. The final predictions are done using task-specific linear projection layers after the InvPT decoder.

2.4.3 Multi-Task Learning Optimisation

In MTL, the model is trained for more than one objective. The most commonly used approach is to calculate the loss for each task using a suitable loss function then compute the weighted sum of all these losses to optimise the model's parameters. The total loss used for back-propagation can be seen in Equation 2.2. Many notable prior work use this method due to its simplicity [77], [78], [85]–[88]. However, using this approach requires tuning these weights to identify the best combination of weights, which is another optimisation step. This is crucial, since the model's performance is sensitive to the weights' values [91].

$$\mathcal{L}_{Total}(W) = \sum_i \theta_i \mathcal{L}_i(W) \tag{2.2}$$

where θ_i is the weight of task i and $\mathcal{L}_i(W)$ is the loss of task i.

The past few years saw some contributions that aim to automate the process of tuning the tasks' weights. The approaches are divided into two categories: weight adaptation and Pareto optimization (PO). Weight adaptation techniques focus on changing the weights during training using a pre-defined heuristics, such as uncertainty estimation [91], gradient normalization [92] and weight average [93]. On the other hand, PO methods aim at formulating the MTL as a multi-objective optimization problem and try to find a Pareto stationary solution [94]–[96].

The work by Kendall et al. (2018) showcases that the intrinsic task-dependent uncertainty provides information about the relative confidence between tasks [91]. The authors propose learnable task weights using a joint likelihood formulation of this uncertainty. In classification tasks, they define their likelihood as a scaled version of the model output through the softmax function $Softmax(\frac{1}{\sigma^2} f^W(x))$. Consequently, they provide an expression for an optimization function that integrates the learnable tasks' weights as follows:

$$\mathcal{L}(W, \sigma_1, \sigma_2) \approx \frac{1}{\sigma_1^2}\mathcal{L}_1(W) + \frac{1}{\sigma_2^2}\mathcal{L}_2(W) + \log \sigma_1 \sigma_2 \tag{2.3}$$

where $\mathcal{L}(W, \sigma_1, \sigma_2)$ is the minimisation objective (back-propagated loss), $\mathcal{L}_1(W)$, $\mathcal{L}_2(W)$ are the task losses and $\sigma_1 \sigma_2$ are the tasks' observation noise parameters.

2.4.4 Multi-Task Learning in Food Image Processing

MTL in food image processing is relatively an unexplored area of research due to the lack of multi-task food datasets. Zhang et al. (2016) proposed dish identifier, cooking-method recognizer and a multi-label ingredient detector that share the low level layers in a CNN [97]. They compared their results against prior work that used handcrafted features and reported 57.25% top-1 accuracy for the dish identification task and 69.5% for the cooking-method recognition. Another work by Ege and Yanai (2018) investigated the role of MTL in food calorie estimation and food dish detection using CNN-based model [98]. Since there is no food dataset annotated with bounding boxes and food calorie information, the authors used two datasets together to train their model. Their work showed that a multi-task model trained to detect food and predict food calorie produces better results with fewer parameters than two sequential models trained on food detection and food calorie estimation. More recently, Liang et al. (2021) proposed a multi-view attention network (MVANet) that is trained on Chinese food recognition, ingredient recognition and recipe generation [99]. The authors developed a a multi-view attention fusion (MVAF) mechanism that allows the extraction and fusion of multiple semantic features from different tasks. The model consists of stacked multi-view attention blocks that process the image representation produced by the convolution layer. Afterwards, three task-specific heads are used to compute the output for each task, and their aggregated loss is used to optimise the model's parameters.

To summarize, there has been a few attempts at applying multi-task learning in food image processing in order to provide more context for the model to distinguish between similar looking food items. However, this area of

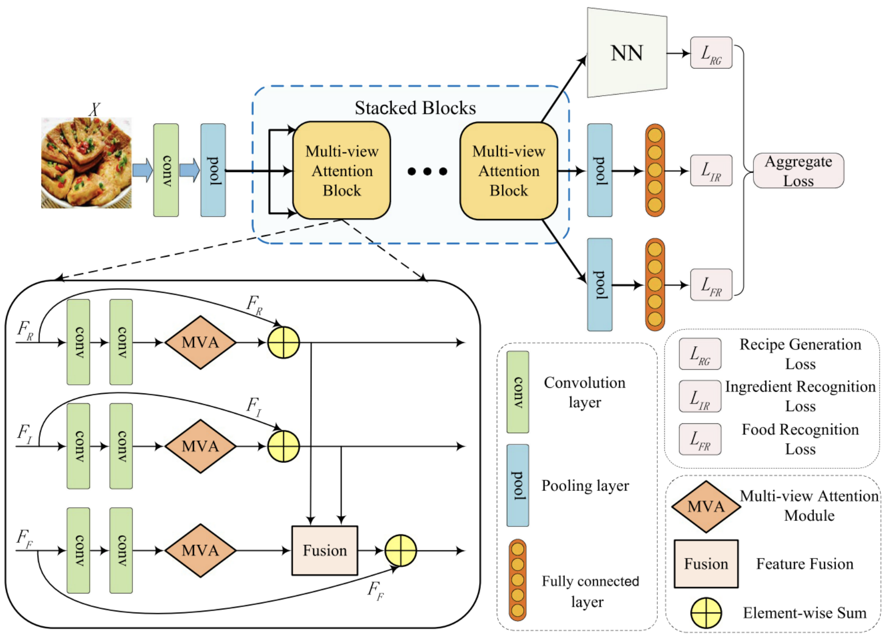


FIGURE 2.10: Multi-view attention network architecture. Image from [99]

research is still under-explored. All the work outlined in the previous section uses CNNs to encode the image features but there has not been a study on how ViT perform in this domain. The self-attention in ViT is a strong tool to provide more global features of food images and has been used successfully for single-task food recognition [60], [63]. Moreover, the studies on multi-task learning for food recognition focus on classification tasks without addressing segmentation. In this thesis, I aim to bridge this gap and provide a a framework that uses related tasks as context to help the model distinguish between food items. In addition, I aim to incorporate classification and segmentation tasks to investigate which tasks benefit from the multi-task learning paradigm.

Chapter 3

Methodology

3.1 Data

In this thesis, I will use Orbisk's own dataset which has 933,207 food waste images manually annotated. The dataset is a comprehensive collection of annotated food waste images that surpass existing benchmark datasets on food image segmentation. It surpasses them in terms of image volume and ingredient coverage as it has 90x the number of images and 7x the number of ingredients. The dataset is verified by Orbisk's internal quality control team that periodically reports any inconsistencies in annotations.

3.1.1 Collection

The images are collected using Orbisk's Orbi as shown in *Figure 1.1*. The camera is triggered using a motion sensor that detects movement in front of the image. Once it's triggered, the camera takes an image with 1920x1440 resolution and the weight from the scale is recorded. The image registration is saved in the cloud database as a JPEG file. After that, the image is passed through the annotation pipeline to obtain the required information from the image. The annotation pipeline first runs the image through Orbisk's AI models to get their predictions. An image confidence model is used to evaluate these predictions and decide whether to trust them or not. If the predictions are not trusted, the image is sent to the manual annotators to manually check the image and correct the AI labels. In this thesis, only manually annotated images are used for the training and testing of the proposed model.

3.1.2 Annotation

Ingredients The most important annotation on the image is the ingredient polygon. Each ingredient of food present in the image is annotated using the polygon coordinates that surround the ingredient along with the ingredient label and category. There are 740 distinct ingredients at Orbisk spread across 21 categories. In total, there are 1,242,333 polygon instances in the dataset. The distribution of ingredients and categories is seen on *Figures* 3.1, 3.2.

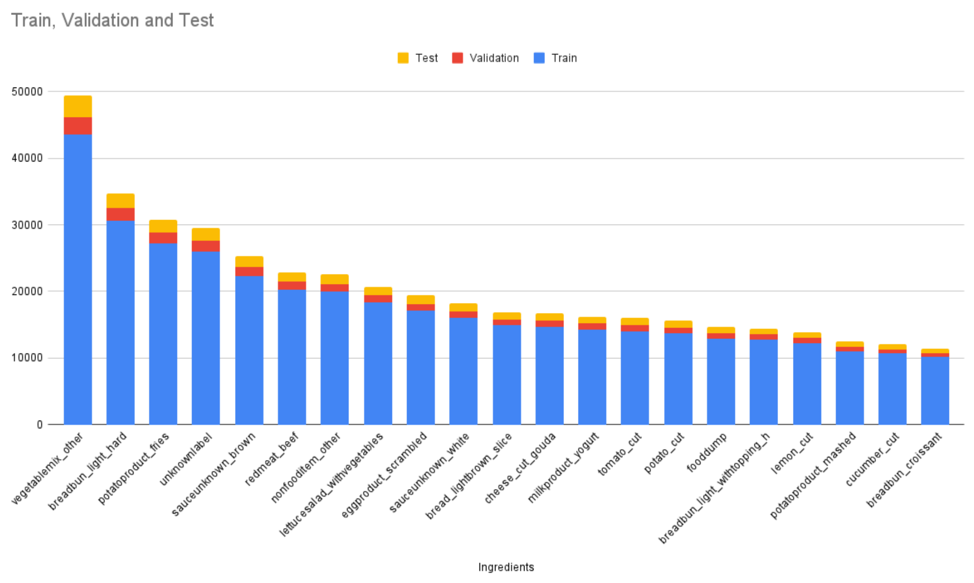


FIGURE 3.1: The distribution of the 21 frequent ingredients.

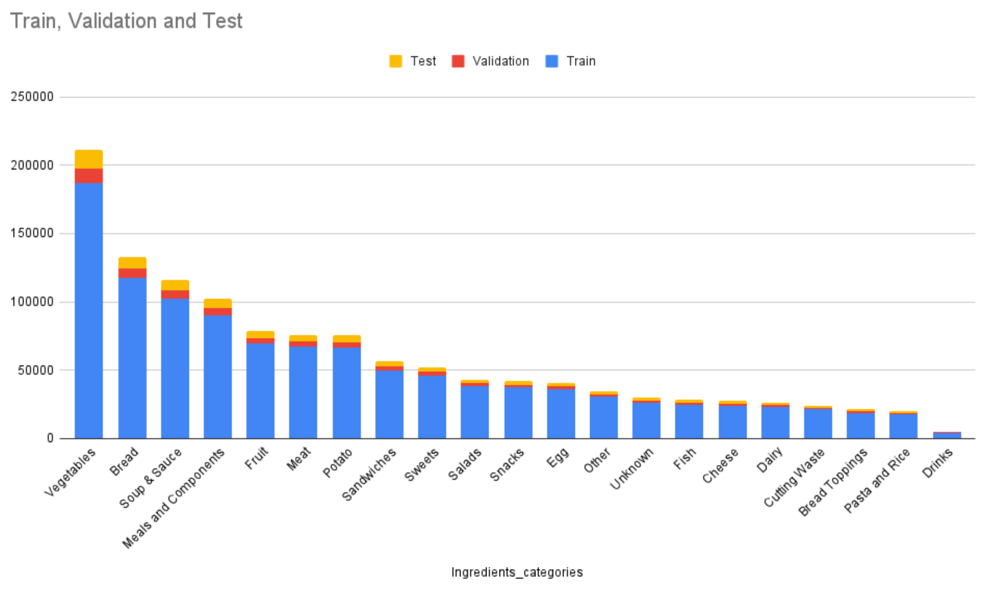


FIGURE 3.2: The distribution of all ingredient categories.

Has-waste In addition to the ingredients' annotation, each image is labeled with a binary has-waste value. This annotation indicates whether the image has waste or not. The has-waste distribution is seen on *Figure* 3.3.

Weight Every image in the dataset gets a weight value indicating the weight of food in the image. The weight values are obtained using Orbisk's weight logic algorithm. The algorithm uses data obtained from the scale and information about the image registration to calculate the weight of the thrown food. The value represents the weight of all food in the image. The distribution of weight is seen on *Figure* 3.4. The weight values are shown in kilograms.

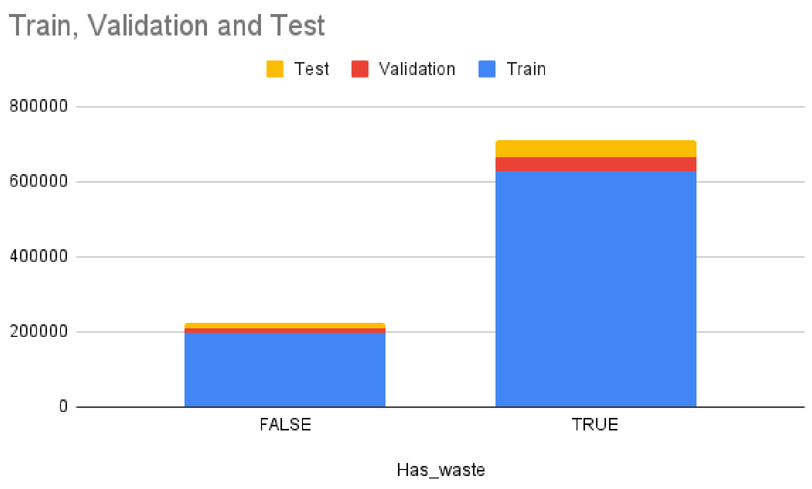


FIGURE 3.3: The distribution of the has-waste annotation.

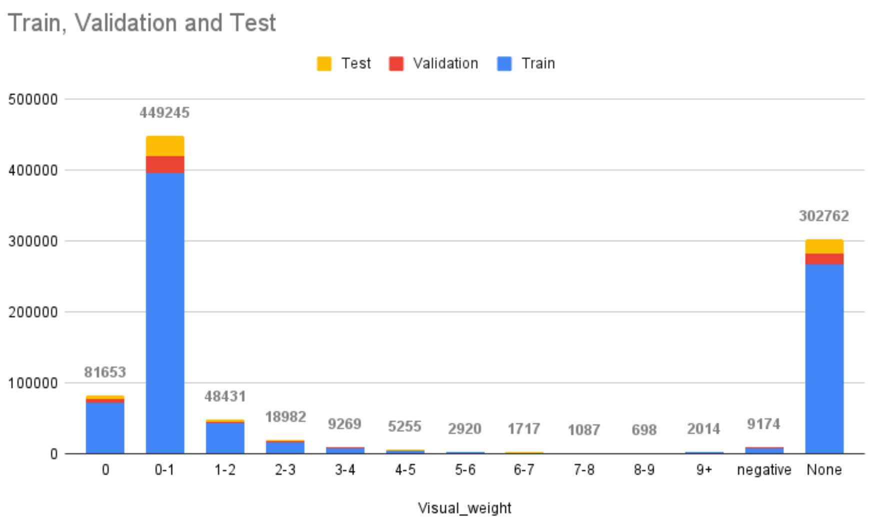


FIGURE 3.4: The distribution of the weight annotation.

Waste-stream To gather more information about the nature of food waste in an image, Orbisk tracks the type of waste thrown under the waste-stream label. This label specifies at which step the food was disposed. The label values and their frequency can be seen on *Figure* 3.5.

Image-quality Moreover, each image is annotated with an image-quality value. This value indicates the quality of the image taken. Poor images are flagged to check with the client whether the hardware solution is optimally placed. The distribution of their values is see on *Figure* 3.6.

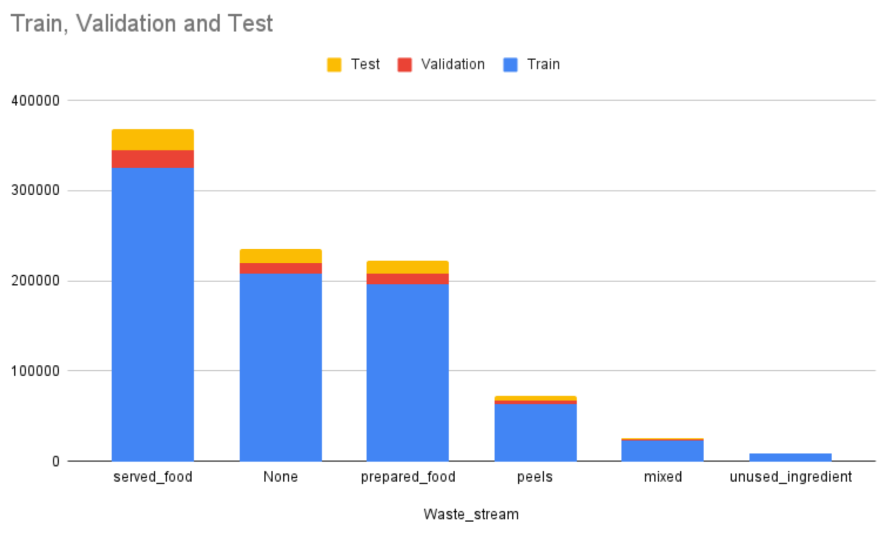


FIGURE 3.5: The distribution of waste-stream label

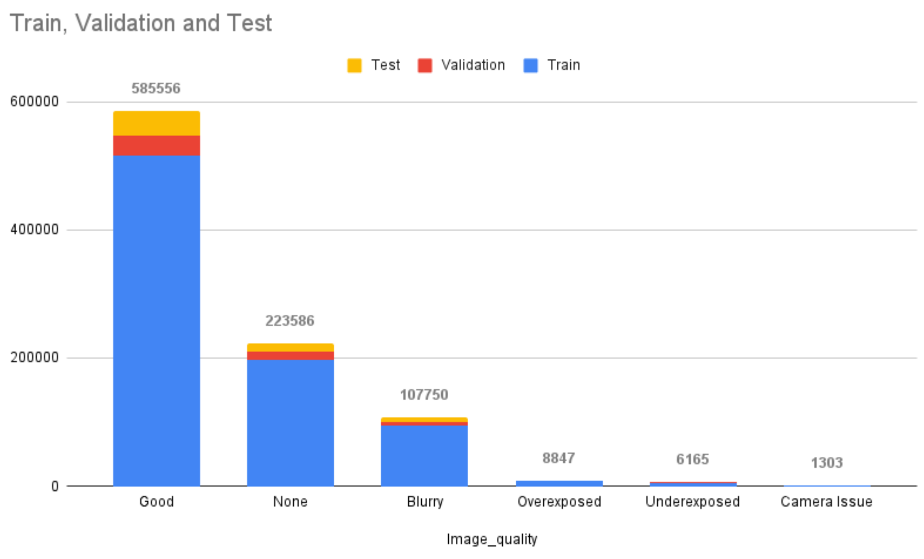


FIGURE 3.6: The distribution of the image-quality annotation.

Container-type Finally, Orbisk keeps track of the container present in the image. The container-type label has 27 distinct values summarizing the different containers the food waste is in. The distribution of this annotation can be seen on *Figure 3.7*.

3.1.3 Pre-processing

The dataset is split into training, validation and test sets. The number of images and masks in every set can be seen in *Table 3.1*. For every image, the ground truth polygons are formatted as a list of (x, y) coordinates tensor while the ingredient labels is a tensor of label indices. For the has-waste, container-type, image-quality and waste-stream annotations, they are formatted as a tensor with the label index. The weight annotation is represented as a tensor with the weight value in kilograms. Each image is normalized using mean values of 123.675, 116.28, 103.53 and standard deviation of 58.395, 57.12, 57.375 for the red, green and blue channels respectively. In addition, the images are resized to 640x480 pixels.

3.2 Mask2Former Models

Cheng et al. (2021) developed *Mask2Former* transformer models to provide a unified framework for all segmentation tasks, namely panoptic, instance and

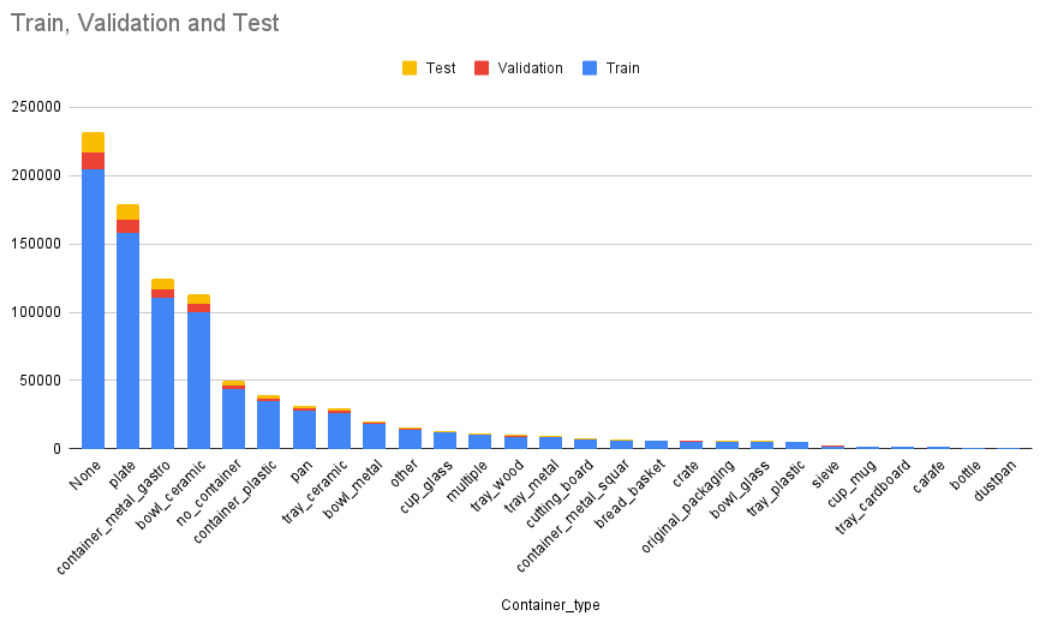


FIGURE 3.7: The distribution of the container-type annotation.

Set	Images	Polygons
Train	823,712	1,096,572
Validation	50,034	66,105
Test	59,461	79,656

TABLE 3.1: Summary of the train, validation and test splits.

segmentation tasks [100]. Standard neural network models focus on solving only one task at a time and do not generalize to address all tasks. For instance, Mask-RCNN [16] and DETR [101] focus on the instance segmentation domain, while FCN [102] addresses semantic segmentation instead. This leads to duplicate optimization of different models to solve similar tasks. *Universal architectures*, on the other hand, aim to develop architectures capable of handling multiple tasks at the same time. MaskFormer [103] and K-net [104] are examples of such networks build to learn multiple segmentation tasks in parallel. These networks still need to be trained separately for different tasks, but their architecture, loss and training procedure can stay the same.

The *Mask2Former* framework uses a mask classification approach to group each pixel into segments by predicting N binary masks with N corresponding labels. Inspired by the DETR architecture, each segment is represented by C dimensional vector ("object query") that can be processed by a transformer decoder that is trained with prediction objective. Hence, the Mask2Former consists of three components. Firstly, a *backbone* is used to obtain low-resolution feature embeddings from an image. Secondly, a *pixel-decoder* gradually upsamples these features to generate the high-resolution embeddings needed for accurate segmentation. Finally, a *Transformer decoder* attends to these features and processes the object queries. The final binary masks are decoded from the per-pixel embeddings with the object queries using a hyperparameter to specify the maximum number of masks per image to predict.

These three components are modular and they can be exchanged with the latest SOTA components. Consequently, there are multiple variants of the *Mask2Former* framework with varying performance measures and complexity. *Table 3.2* summarizes the different variants of *Mask2Former*.

The *Mask2Former* makes a significant contribution to the transformer decoder in order to improve the accuracy. It uses a masked-attention layer within each decoder block to extract localized features within the foreground region of the predicted mask for each each query. This layer replaces the normal

Backbone	**Epochs**	**mAP**	**Parameters**	**FLOPs**
ResNet-50 [105]	50	43.7	44M	226G
ResNet-101 [105]	50	44.2	63M	293G
Swin-T [61]	50	45.0	47M	232G
Swin-S [61]	50	46.3	69M	313G
Swin-B [61]	50	46.7	107M	466G

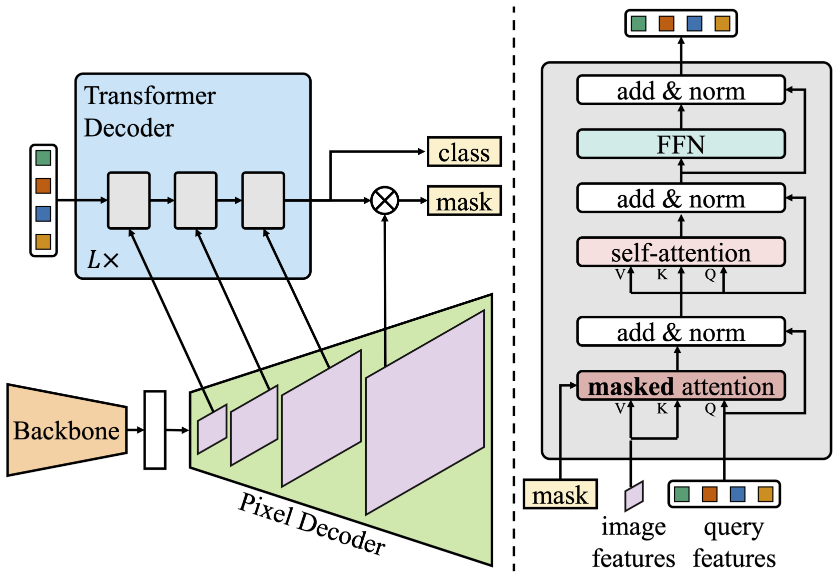
TABLE 3.2: Summary of *Mask2Former* variants on the COCO dataset [106]. The mean average precision metric (mAP) is described in the following section.

cross-attention layer that attends to the full feature map and it showcased a significant improvement across all segmentation tasks. An overview of the *Mask2Former* architecture can be seen in *Figure 3.8a*.

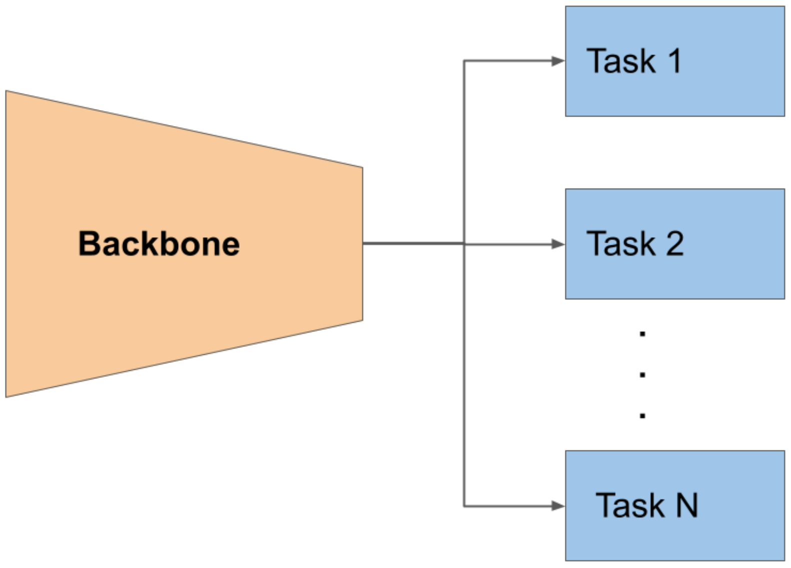
3.2.1 FoodWasteAI Model

In this thesis, I build upon the *Mask2Former* architecture, and modify it to address all tasks Orbisk. More precisely, the FoodWasteAI architecture uses the same meta architecture as *Mask2Former* with additional heads connected to the backbone for each task. The backbone provides a low-resolution feature vector that is decoded by each task's head to arrive at the required prediction. The model is trained in an end-to-end fashion using the annotation data of each task.

Three post-processing steps are used to refine the ingredient masks produced by the model and eliminate the false-positive predictions. The first step is filtering out the low-scoring masks using a score threshold α_{score}. Any masks with a lower confidence score than the threshold are removed from the predictions. The second step is joining overlapping polygons that share the same label. This way, if there two overlapping masks with the same label, they are merged and predicted as one mask instead of multiple. The joining threshold is called α_{join}. Finally, a non-maximum suppression (NMS) algorithm is used to suppress overlapping polygons with lower confidence score. The NMS overlapping threshold is called α_{nms}.



(A) *Mask2Former* architecture Image from [100].



(B) FoodWasteAI architecture.

3.3 Evaluation Metrics

This thesis addresses three research sub-questions that require the use of different metrics. To evaluate the model's performance on the different tasks, the common metrics reported in academic papers are used. In addition, the models are compared using business metrics to evaluate their performance in the test simulation.

3.3.1 Academic Metrics

Classification In this thesis, I use top-1 and top-5 accuracy to report the performance of all classification tasks. Top-1 accuracy measures the amount of predictions that are exactly the target class [107]. This metric is the most commonly reported metric in image classification research [25], [26], [61].

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

where TP is true positive, meaning that the model correctly predicts the positive class. TN is the true negative which means the model correctly predicted the negative class. FP and FN are the false positive and negative respectively, which means when the model falsely predicts the positive or negative classes.

Regression For the regression task, I use the weighted mean absolute percentage error as described in *Equation* 3.2. This metric is used to evaluate the performance of regression models by calculating the absolute error relative to the ground-truth values. Given the task at hand, the weighted error is more informative regarding the relative error in weight that the model makes.

$$\text{WMAPE} = \frac{\sum_{t=1}^{n} |y_t - \hat{y}_t|}{\sum_{t=1}^{n} |y_t|} \tag{3.2}$$

where y_t is the ground-truth value and $\hat{y}_t$ is the predicted value.

Instance Segmentation In this thesis, I report the COCO mean average precision (mAP) for the instance segmentation tasks. COCO mAP measures the average precision of detection across a range of intersection over union (IoU) values. More precisely, the metric calculates the average precision (AP) across all classes and 10 IoU thresholds in the range of 0.5 - 0.95 with a step size of 0.05. The IoU measures the overlap between two masks or bounding boxes as shown in *Equation* 3.4. The mAP metric is widely used in instance segmentation research [100], [101], [103].

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.3}$$

where TP is true positive and FP is false positive.

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} \tag{3.4}$$

where A is the predicted mask and B is the ground-truth mask.

$$\text{mAP} = \frac{1}{10} \sum_{i=0}^{9} AP@(0.5 + i \cdot 0.05) \tag{3.5}$$

where AP is the AP across all classes.

Model size To measure the model size, the number of parameters in the model will be reported. Number of parameters in the model provides information about the efficiency and complexity of the model. This metric is used in research to showcase the model's size and speed [27], [58]. Therefore, it helps in estimating the costs of hosting the model on the cloud and edge devices. This metric is computed by counting the total number of parameters in the model.

3.3.2 Business Metrics

To evaluate the performance of the model in Orbisk's pipeline, the overall picture accuracy will be reported. Picture accuracy (PA) represents the percentage of images in which the model predicted all polygons and ingredients correct. This is calculated by counting the number of images for which the model's predictions were correct on the polygon and ingredient level then dividing it by the total number of images the model received. PA is valuable to Orbisk as it allows them to understand how accurate the model is on an image level.

In addition to PA, the food picture accuracy metric will be reported. This metric is similar to PA but it involves only images with food waste. All images without food are discarded and not used in this metric. As a result, it measure how accurate the model predictions are when there's food waste on the image.

$$\text{PA} = \frac{I}{N} \tag{3.6}$$

where I is the number of images with all polygons and ingredients correctly predicted (images with no waste still count), and N is the total number of images received.

$$\text{FPA} = \frac{\hat{I}}{\hat{N}} \tag{3.7}$$

where $\hat{I}$ is the number of images with all polygons and ingredients correct, and $\hat{N}$ is the total number of images with food received.

3.4 Software

In this thesis, the code was done in python. Developing the models is done with the Openmmlab framework which is built on top of PyTorch. More specifically, I utilized their mmdetection [108] and mmpretrain [109] libraries. The training configurations are declared using their config-file structure. The training runs are logged to mlflow [110] which is hosted on Orbisk's cloud services. Data visualization is done using the fiftyone library [111]. The models are trained on Orbisk's servers, which has three Nvidia A5000 GPUs, in addition to one Nvidia RTX 2090 GPU.

Chapter 4

Experiments

This section outlines the experiments setup done in this thesis. Firstly, I describe how the single-task models are trained and evaluated. This experiment addresses the first research sub-question. Secondly, I outline the training and evaluation procedure for two FoodWasteAI variants. The first variant (FoodWasteAI-Cls) combines all tasks except ingredient segmentation into one model. The second variant (FoodWasteAI-All) combines all tasks into one model trained end-to-end. Finally, I compare the FoodWasteAI-All model to Orbisk's current setup in a test simulation.

The objective of these experiments is to understand how effective multi-task learning is in food waste detection and what are its shortcomings. The experiments highlight which tasks benefit from the multi-task paradigm and whether all tasks can be efficiently handled by one model. With that in mind, all the models trained in the experiments have the same architecture described in *Figure 3.8b*. This is to ensure a fair comparison between the single and multi-task models trained in these experiments. In the baseline experiments, single-task models are trained to understand how well the model performs on each task separately and to analyse the benefits of the multi-task learning experiments. The main difference in experiments are which heads are switched on and used for both training and validation.

The backbone used for all models is the Swin-T [61] model pre-trained on the ImageNet-1k dataset [112]. This backbone shows incredible results across various applications in classification and segmentation tasks and is relatively efficient to train. For the task-heads, a single linear layer followed by a softmax activation is used for all classification and regression tasks. As for the segmentation head, the pixel and transformer decoders used in Mask2Former [100] are used to predict the masks. A summary of all models used in this thesis is highlighted in *Table 4.1*.

Model ID	**Tasks**	**Training time**	**No. of parameters**
FoodWasteAI-HW	Has-waste	36h	27.52M
FoodWasteAI-CT	Container-type	36h	27.53M
FoodWasteAI-WS	Waste-stream	36h	27.53M
FoodWasteAI-IQ	Image-Quality	36h	27.53M
FoodWasteAI-VW	Visual-weight	36h	27.52M
FoodWasteAI-IS	Ingredient-segmentation	146h	47.65M
FoodWasteAI-Cls	All Cls. and Reg. tasks	36h	27.56M
FoodWasteAI-All	All tasks	156h	47.69M
Orbisk's Setup	All tasks	204h	73.78M

TABLE 4.1: Summary of all models trained. The FoodWasteAI-Cls combines all classification and regression tasks. Note that Orbisk's setup is comprised of two models.

The experiments use the same training loop configuration to train the model. The weight-decay Adam (AdamW) optimizer is used to update the models' weights in back-propagation. A learning rate of 0.0001 and a weight decay of 0.05 are used. A linear parameter-scheduler is used in the first 500 iterations to warm up the model with a start factor of 0.001. Afterwards, the multi-step learning-rate scheduler is used for the remainder of the training with a gamma factor of 0.1 at 90% and 95% of the training iterations. The models are trained for 123,000 iterations each with a batch size 15.

To handle the high memory consumption of the masks needed in training the segmentation head, a distributed training with three Nvidia GeForce A5000 GPUs are used. Each GPU handles a batch size of 5. Every 12,300 iterations, the validation set is used to obtain the model's performance thus far. In total, 10 validation logs are performed during a full training. All training and validation performances are logged to the Mlflow server hosted on Orbisk's cloud computing resources.

Due to the dynamic nature of images at Orbisk, the training data is fed to the model using an infinite sampler strategy. The training data is shuffled then fed to the model and when all training data are consumed, they are reshuffled and fed again. Since the data at Orbisk is continuously changing, this strategy ensures that the exact number of training data can be specified using a batch size and an iteration number. As mentioned above, the number of iterations is 123,000 and a batch size of 15 (5x3) meaning that the training phase uses 1,845,000 samples. On the other hand, the validation set is fed using a default sampling strategy where all the validation data is passed

Model	Batch Size	Learning Rate	Iterations	Heads
Single-task	5x3	0.0001	123,000	1
FoodWasteAI-Cls	5x3	0.0001	123,000	5
FoodWasteAI-All	5x3	0.0001	123,000	6

TABLE 4.2: Summary of hyper-parameters used in the experiments. Note that the batch size indicates the number of images per GPU x the number of GPUs used.

through the model once without shuffling.

4.1 Single-Task Models

Due to the lack of food waste detection in machine learning, there is no comparable baseline performance available. Hence, this experiment serves as a baseline for the FoodWasteAI framework proposed in this thesis. The experiment aims to showcase the performance of the model on each task separately. For this, the validation set is used to evaluate each model's performance. This should give a point of reference when analysing the FoodWasteAI-All performance. Combining all tasks in an end-to-end training may yield a better or worse performance than the single-task training.

4.1.1 Classification

All model variants in this category are trained using the categorical cross-entropy loss which is defined in *Appendix A*. This loss is used in many classification problem to quantify the error in the model's prediction and update its weights.

FoodWasteAI-HW This model is trained on the has-waste task, which is a binary classification problem to classify whether there is food in the image or not. This task is trained using the has-waste annotation of each image as described in *Section 3.1.2*.

FoodWasteAI-CT This variant is trained on the container-type task, which is a 27-class classification problem that identifies the type of container present in an image. The container-type annotations are used to train this model.

FoodWasteAI-WS The waste-stream task has six classes which summarize the type of waste visible in the image. Similarly, the waste-stream annotations, as described in *Section 3.1.2* is used to calculate the loss and back-propagate it through the model.

FoodWasteAI-IQ Finally, the last classification task is the image-quality. This is a six-class problem which indicates the quality of the image. The image-quality annotations are used to calculate the cross-entropy loss which is used to update the model's weights.

4.1.2 Regression

FoodWasteAI-VW The visual-weight task is a regression problem focusing on predicting the weight of the food waste in an image. This task is trained using the weight annotations of the image as described in *Section 3.1.2*. The task head is a linear layer initialised using a normal distribution. The loss from this head is calculated using the smooth l1 loss function as described in *Appendix A*. Only this loss is back-propagated through the model to update its weights.

4.1.3 Segmentation

FoodWasteAI-IS The ingredient segmentation task involves predicting instance masks with a corresponding label for each ingredient in the image. For this, the ingredient annotations described in *Section 3.1.2* are used to train the model. The architecture and hyper-parameters of the pixel decoder and transformer decoder used in the Mask2Former paper are left unchanged. The post-processing steps outlined in *Section 3.2.1* are omitted in this experiment. The model is trained using a combination of mask loss and classification loss. The mask loss uses binary cross-entropy and dice loss as described in *Equation 4.1*. The total loss adds the mask loss to a weighted classification loss as described in *Equation 4.2*.

$$\mathcal{L}_{mask} = \lambda_{ce}\mathcal{L}_{ce} + \lambda_{dice}\mathcal{L}_{dice} \tag{4.1}$$

where λ_{ce} = 5.0 and λ_{dice} = 5.0.

$$\mathcal{L}_{total} = \mathcal{L}_{mask} + \lambda_{cls}\mathcal{L}_{cls} \tag{4.2}$$

where $\lambda_{cls} = 2.0$.

4.2 FoodWasteAI

This section outlines the two version of FoodWasteAI models that are trained on multiple tasks. The first variant is trained on the classification and regression tasks only, while the second variant is trained on all tasks. Both variants use the same architecture defined in *Figure* 3.8b with the appropriate heads turned on. The validation set is used to evaluate the models' performance. The loss used to train each model is a linear combination of each task's loss, which were described in the previous subsection. More specifically, the total loss for the model is calculated as shown in *Equation* 4.3

$$\mathcal{L}_{total} = \sum_{i=0}^{n} \lambda_i \mathcal{L}_i \tag{4.3}$$

where $\lambda_i = 1.0$ and $\mathcal{L}_i$ is the loss for task i.

4.2.1 FoodWasteAI-Cls

This model variant trains on all tasks except the segmentation one. This is equivalent to Orbisk's multi-task model which focuses on predicting the appropriate label on an image level. The importance of this model is that it provides a glimpse on how effective multi-task learning is for this set of tasks as compared to the single-task models. Moreover, it showcases whether the Swin transformer backbone has the capacity to handle more than one task concurrently. The model uses the same backbone as the single-task models, with the appropriate task heads enabled. The performance of each task is evaluated using the corresponding metric as discussed in *Section* 3.3.

4.2.2 FoodWasteAI-All

This model combines all tasks into one model. Similar to the previous variant, this variant adopts the same architecture with all task heads enabled. The three post-processing steps are not used in this experiment. The performance of this model is evaluated using the evaluation metric for each task type as outlined in *Section* 3.3.

4.3 FoodWasteAI v.s. Orbisk's Setup

The last experiment runs the FoodWasteAI-All model in a test experiment in order to simulate its performance in a production environment. The model is used with the test set, which has not been seen by the model before, in order to understand the model's performance on an image and polygon level. In addition, the model is compared to Orbisk's current AI models to understand the limitations of such model in a real-world application. The picture accuracy (PA) and food picture accuracy (FPA), described in *Section* 3.3, are used as the main evaluation metrics.

4.4 Ablation Studies

This section outlines the ablation studies conducted in this thesis. The first one investigate the effect of adding augmentations to the training set before feeding it to the model. The second study investigated the effect of the post-processing steps used to refine the masks predictions. A list of the these experiments are found in *Table* 4.3.

4.4.1 Augmentations

This study investigates the effects of augmentations on the model's performance. Models V0-V5 are used in this study. Due to the long time it takes

ID	Post-processing	Augmentations	Dataset
V0	-	-	Subset
V1	-	Contrast	Subset
V2	-	Brightness	Subset
V3	-	Rotation	Subset
V4	-	Flip	Subset
V5	-	Contrast & Brightness	Subset
V6	-	-	Full
V7	Filtering	-	Full
V8	Join & NMS	-	Full
V9	Filtering & Join	-	Full
V10	Filtering & NMS	-	Full
V11	Filtering, Join & NMS	-	Full

TABLE 4.3: Summary of ablation experiments conducted. The "Dataset" column specifies whether the full dataset or a subset of it is used. The subset dataset is defined in the *Section* 4.4.1.

each model to train, a lower number of iterations is used for the training. More specifically, each model variant is trained for 9,000 iterations and evaluated on 5,000 validation samples. All other hyper-parameters are fixed according to the model's description in the above section. On average, it takes each model 4 hours to train and validate.

The contrast and brightness augmentations shifts their value by 20% either positively or negatively. The rotation augmentation randomly chooses an angle between $-45°$ - $+45°$. The flipping augmentation flips the image either vertically, horizontally or both. The probability of each augmentation is 0.33.

4.4.2 Mask Post-Processing

This study analysis the effectiveness of the post-processing steps to refine the predicted masks for the ingredient segmentation task. The Mask2Former model produces 100 predictions per image, which means that there are a lot of false-positive predictions. To remove them, the three post-processing steps defined in *Section 3.2.1* are used. Each step eliminates a portion of the false-positive predictions, but may accidentally remove some correct masks. To understand whether each step is effective in increasing the model's performance, variants V6-V10 are used to predict masks on the validation set. Their performance is analyzed using the picture accuracy (PA) and food picture accuracy metrics (FPA). In addition, the filtering step is analysed using score thresholds in the 0.1-0.9 range to understand the sensitivity of the masks' confidence scores.

Chapter 5

Results

This section presents the results of the experiments conducted in this thesis. Firstly, I present the performance of single-task models trained on each task and how it compares to the multi-task models variants. Moreover, I delve deeper in analyzing the performance of the FoodWasteAI model and showcase quantitative and qualitative results of the model's prediction. Afterwards, I present the results of the test simulation on the FoodWasteAI and Orbisk's current setup. Finally, the results of the ablation studies are reported in the final section. *Table 5.1* summarizes the performance of all models trained in this thesis on the validation set.

Model	**VW**	**WS**	**CT**	**HW**	**IQ**	**IS**
Single-Task	0.71	87.50	**86.34**	**94.43**	**90.64**	27.20
FoodWasteAI-Cls	**0.69**	87.59	85.66	94.20	90.48	-
FoodWasteAI-All	0.72	**87.67**	85.16	94.23	90.29	27.30
FoodWasteAI-All+	0.72	87.59	85.07	94.19	90.11	**27.60**

TABLE 5.1: Summary of FoodWasteAI results and the baseline single-task models. Each model is evaluated on the Visual-Weight (VW), Waste-Stream (WS), Container-Type (CT), Has-Waste (HW), Image-Quality (IQ) and Ingredient-Segmentation (IS) tasks. The Visual weight task is evaluated using the WMAPE (*Equation 3.2*), the ingredient segmentation task uses mAP (*Equation 3.5*) and the other tasks use accuracy (*Equation 3.1*). The "+" version is with augmentations on the training set.

5.1 Single-Task Models

The single-task models are trained for 123,000 iterations and their training and validation performance is recorded 10 times throughout the training run. The academic metrics are used to evaluate these models and compare them to the multi-task variants. The complete history of the single-task models can be found in *Appendix* B.

As we can see in *Table* 5.1, the single-task models performed decently on all tasks. Most notably, they outperformed the multi-task variants on the *Container-type, Has-waste and Image-quality* tasks.

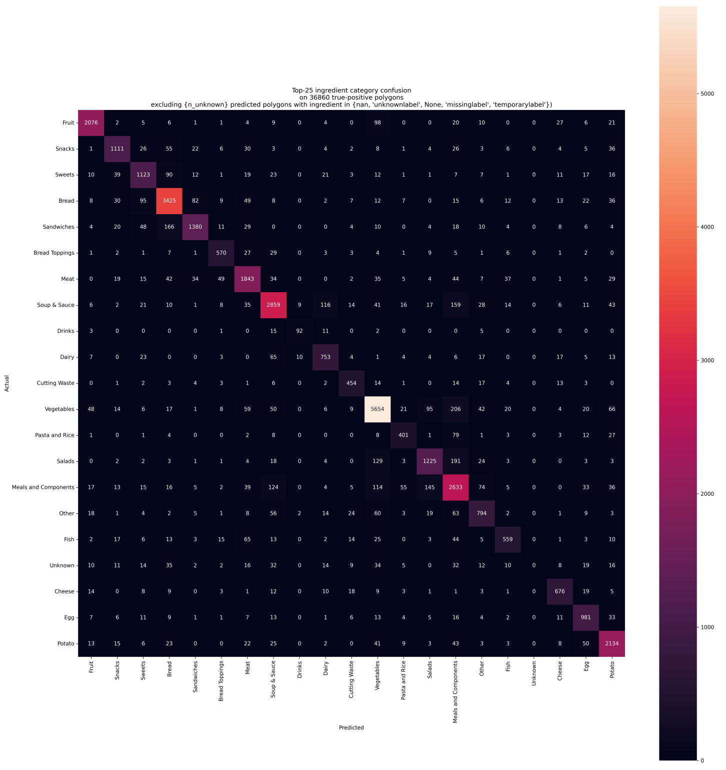
5.2 FoodWasteAI

This section dives deeper in the results of the FoodWasteAI model. More specifically, an analysis of the ingredient segmentation task is performed to understand how well the model performs on the ingredients' predictions.

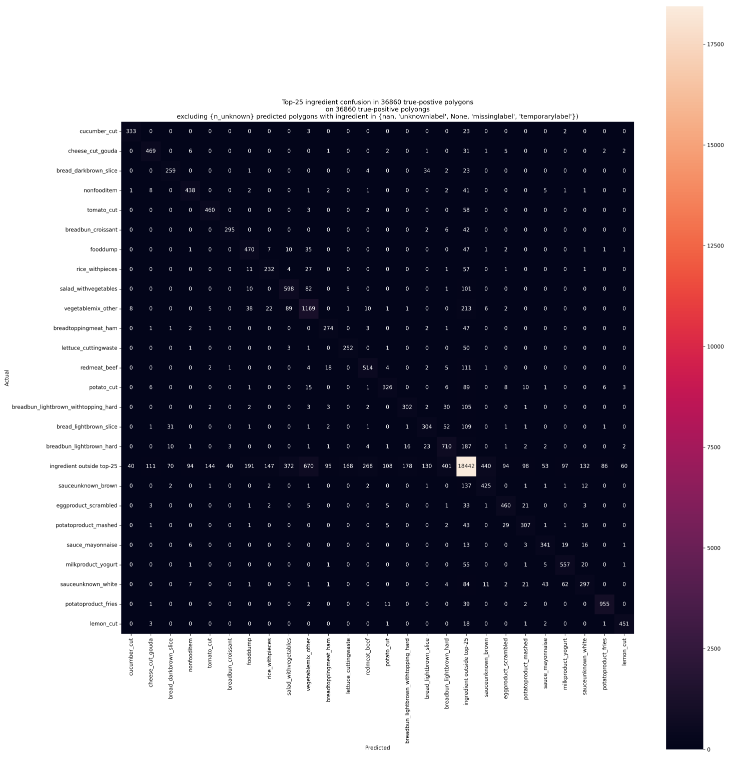
5.2.1 Quantitative results

The confusion matrix for both ingredient labels and categories can be seen in *Figures* 5.1a, 5.1b. The figures show all 21 categories and the top-25 ingredient labels. The remaining labels are squashed into on label called "ingredients-outside-top-25". The category confusion matrix shows that the model is able to grasp the difference between the different category values. The model's prediction match the ground-truth categories with high frequency as can be seen on the diagonal values. The mismatch between the categories is not substantial but evident in some categories. For instance, "Meals and Components" is predicted wrongly instead of "Salads", "Vegetables" and "Soup & Sauce".

On the hand, the ingredient labels are harder to analyze due to the large number of classes. However, upon analyzing the top-25 classes in terms of their frequency, it is shown that the model is good at differentiating them apart, with most non-diagonal cells having a value of 0 or 1. Nonetheless, there are some labels that have relatively large frequency of mismatch. The "sauceunknown_white" is misclassified as "potatoproduct_mashed", "sauce_mayonnaise" and "milkproduct_yoghurt" with frequencies of 21, 43 and 62 respectively.



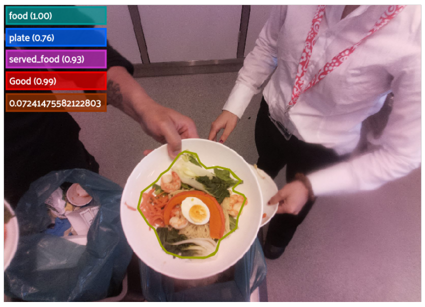
(A) Ingredient categories.



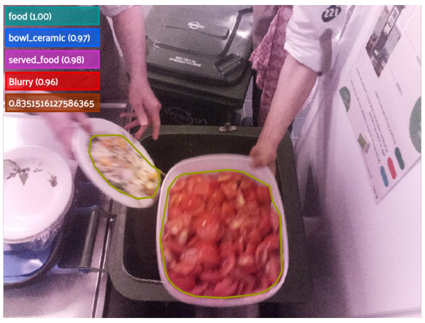
(B) Ingredient labels.

FIGURE 5.1: Confusion matrix for ingredients' categories and labels on the validation set.

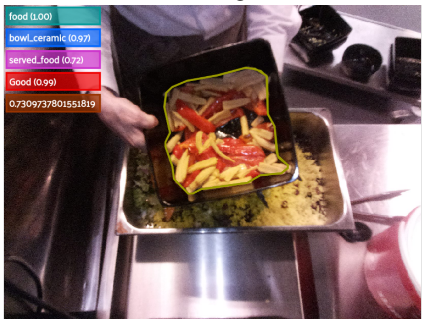
5.2.2 Qualitative results



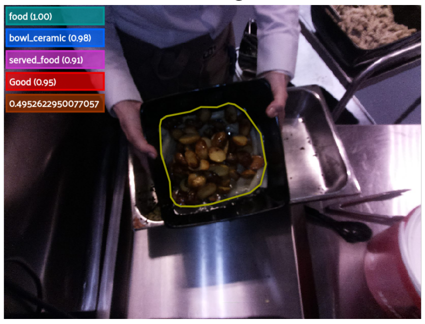
(A) Image 1



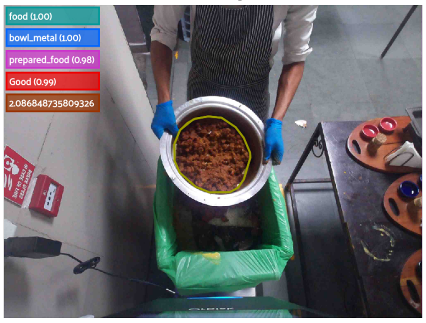
(B) Image 2



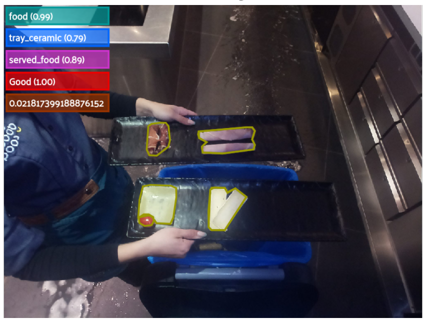
(C) Image 3



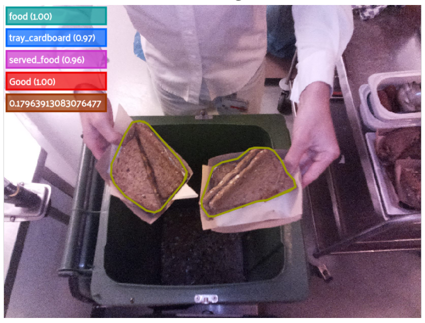
(D) Image 4



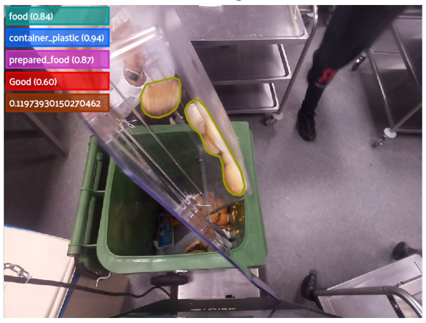
(E) Image 5



(F) Image 6



(G) Image 7



(H) Image 8

FIGURE 5.2: Qualitative results of FoodWasteAI on validation set. The prediction of the other tasks is found in the top-left corner. For each task, the confidence score is in brackets next to the predicted label. The green box showcases the has-waste label, the blue box shows the contain-type, the purple one is for the waste-stream task, the red is for the image-quality and the brown showcases the visual weight in log-scale.

5.3 FoodWasteAI v.s. Orbisk's Setup

This section presents the results of the test simulation conducted on both the FoodWasteAI model and Orbisk's current models. This experiment simulates a production environment where the test set is used to understand how well the model performs when deployed. The business metrics are used to compare both setups.

5.3.1 Quantitative Analysis

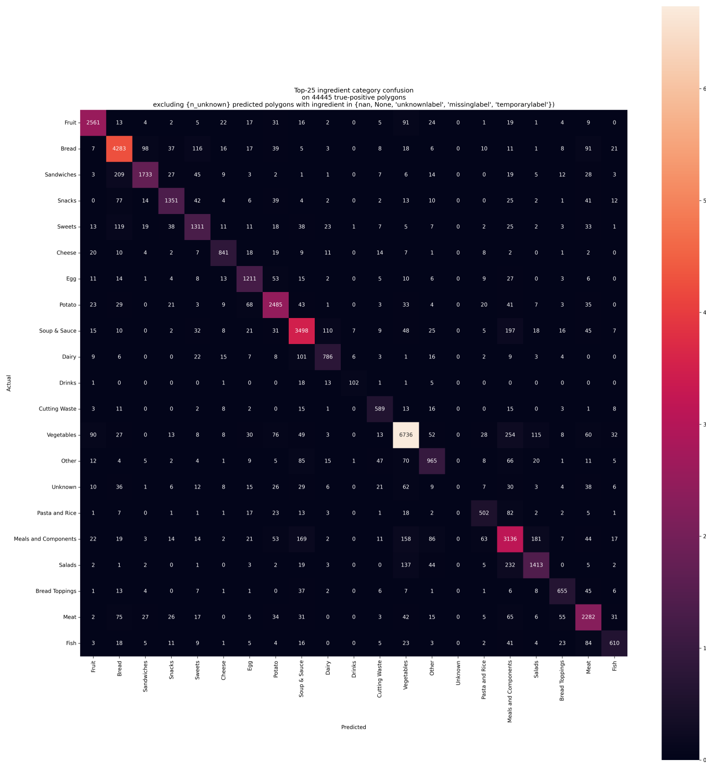
To understand the performance of the model on the segmentation task, the confusion matrix on the top-25 labels and categories is computed. The remaining ingredients are squashed into on label called "ingredients-outside-top-25". *Figures 5.3a, 5.3b* showcase these results. They show that the model is able to distinguish well between the different ingredient categories. This is shown by the high numbers across the diagonal, which indicate a match between the predicted and actual category. Moreover, the other cells have a relatively low number which further suggests that the model's predictions are accurate on the category level. However, there seems to be some categories that the model predicts wrong more than others. For instance, the "Meals and Components" category is predicted 254 times where the actual category was "Vegetables". In addition, there seems to be a correlation between the "Soup & Sauce" and "Meals and Components" category as the model wrongly mixes them with each other.

On the other hand, the ingredient labels matrix showcases similar results. The non-diagonal values are mostly either 0 or 1, which indicates that there is low confusion among the different label values. The most confused labels seem to be "bread_lightbrown_slice" and "bread_darkbrown_slice". Moreover, "vegetablemix_cuttingwaste" appears to be mistaken for "fooddump", "vegetablemix_other" and "salad_withvegetables". However, this is understandable due to the high visual similarity between these labels.

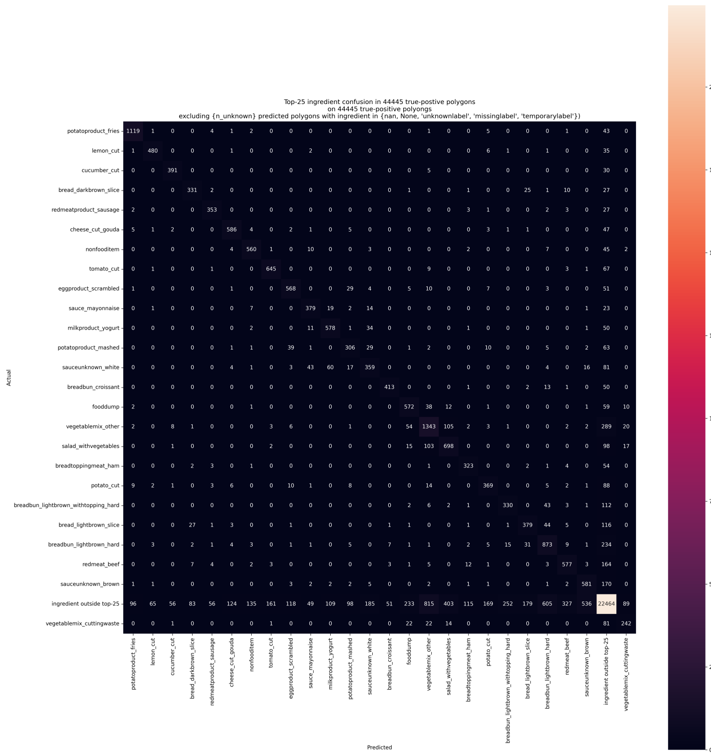
Model	**Picture Accuracy**	**Food Picture Accuracy**	**Ingredient Accuracy**	**Polygon Precision**	**Polygon Recall**
Orbisk setup	40.6	26.6	59.1	74.4	62.2
FoodWasteAI	43.7	30.7	67.7	62.6	60.4

TABLE 5.2: Results of the FoodWasteAI model and Orbisk's current setup in the test simulation.

Figures 5.4a, 5.4b present the precision of true-positive polygons' ingredients and categories. As can be seen, the "Unkown" category is the worst performing one. This is further shown in the precision figure of the ingredient labels as the "unkownlabel" is the worst performing label.

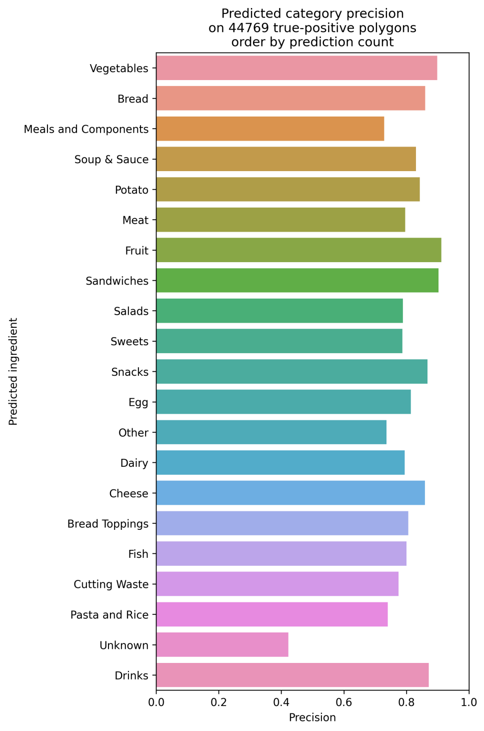


(A) Ingredient categories.

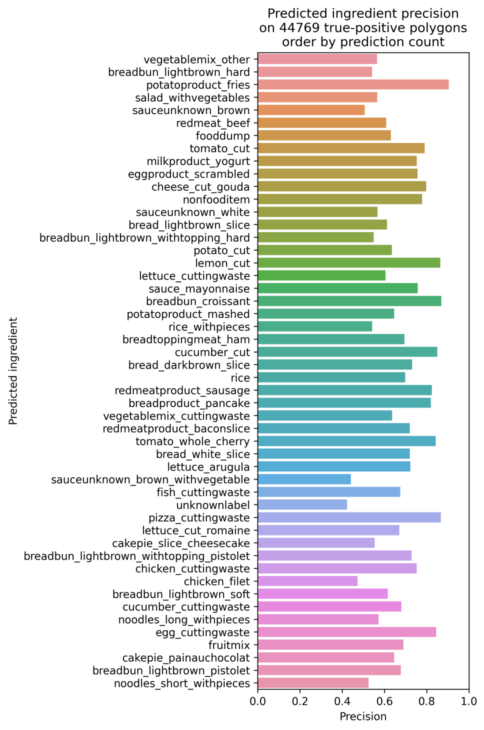


(B) Ingredient labels.

FIGURE 5.3: Confusion matrix for ingredients' categories and labels on the test set.



(A) Ingredient categories.

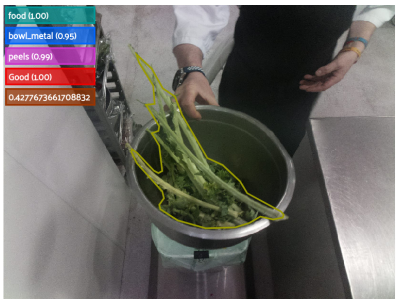


(B) Ingredient labels.

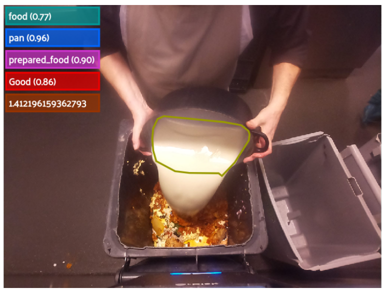
FIGURE 5.4: Precision of ingredients' categories and labels on the test set.

5.3.2 Qualitative Analysis

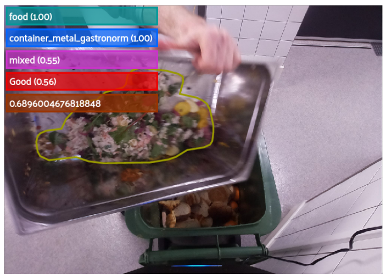
This subsection presents the FoodWasteAI predictions on eight images from the test set.



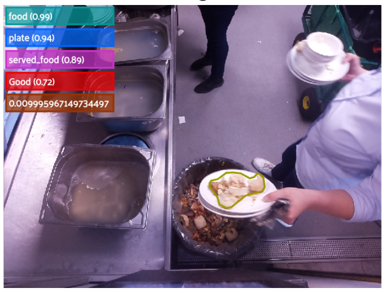
(A) Image 1



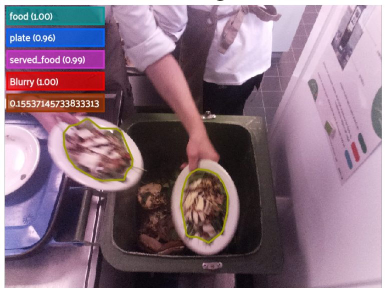
(B) Image 2



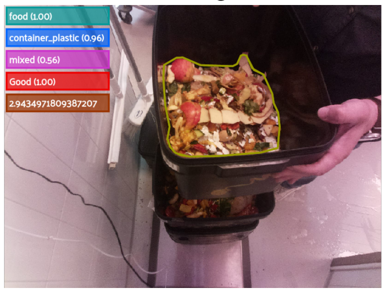
(C) Image 3



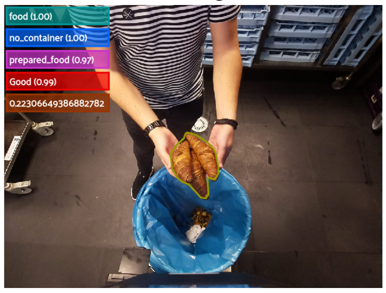
(D) Image 4



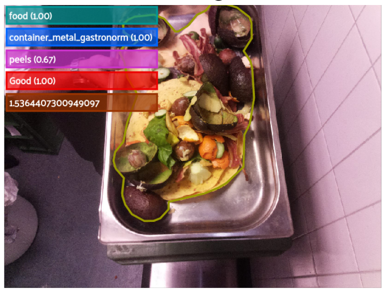
(E) Image 5



(F) Image 6



(G) Image 7



(H) Image 8

FIGURE 5.5: Qualitative results of FoodWasteAI on test set. The prediction of the other tasks is found in the top-left corner. For each task, the confidence score is in brackets next to the predicted label. The green box showcases the has-waste label, the blue box shows the contain-type, the purple one is for the waste-stream task, the red is for the image-quality and the brown showcases the visual weight in log-scale.

5.4 Ablation Studies

This section showcases the results of the ablation studies conducted in this thesis. As mentioned in the previous section, there are two main ablation studies. The first one investigates which augmentation should be applied during training. The second experiment validates the effectiveness of the three post-processing techniques used to refine the mask predictions.

5.4.1 Augmentations

There are 4 augmentation techniques used in this study, namely, brightness-shift, contrast-shift, rotation and flipping. A variant without any augmentations is added for reference. The results of the augmentation experiments can be found in *Table* 5.3. The results show that the *Brightness* and *Contrast* augmentations produced the best results. The *Brightness* augmentation yielded better results on the Visual-Weight and Container-Type tasks. On the other hand, the *Contrast* augmentation produced better results on the Waste-Stream, Has-Waste and Ingredient-Segmentation tasks.

Upon combining both augmentation, the results show that the performance deteriorates on all tasks. Most notably, the Instance-Segmentation task scores much lower than it does with Contrast or Brightness augmentation alone. Therefore, only Contrast augmentation is used to train the main model using the full dataset.

ID	Augmentation	VW	WS	CT	HW	IQ	IS
V0	No Augmentations	0.84	75.94	61.62	86.46	85.89	5.40
V1	Contrast	0.84	**76.29**	62.36	87.09	86.17	**5.70**
V2	Brightness	**0.83**	75.25	**62.70**	86.64	86.28	5.20
V3	Rotation	0.85	74.37	58.58	86.03	84.58	3.30
V4	Flip	0.84	75.22	60.80	86.68	**86.41**	5.10
V4	Brightness & Contrast	**0.83**	75.81	62.62	**87.27**	85.70	4.80

TABLE 5.3: Summary of augmentation results on a subset of the dataset. The FoodWasteAI-All model is evaluated on the Visual-Weight (VW), Waste-Stream (WS), Container-Type (CT), Has-Waste (HW), Image-Quality (IQ) and Ingredient-Segmentation (IS) tasks. The Visual weight task is evaluated using the WMAPE (*Equation* 3.2), the ingredient segmentation task uses mAP (*Equation* 3.5) and the other tasks use accuracy (*Equation* 3.1).

5.4.2 Mask Post-processing

Mask post-processing aims to filter out bad predictions from good ones. The FoodWasteAI model predicts 100 masks with corresponding label for each image. To obtain good predictions, a score filtering process takes place, followed by joining overlapping masks with the same label. Finally, a non-maximum suppression algorithm suppresses overlapping masks with the lower score. This subsection investigates the effectiveness of these three steps and analysis the sensitivity of the score threshold used for filtering out predictions.

Table 5.4 showcases the performance of the model on the validation set using the different post-processing strategies. The V6 experiment acts as a control experiment where no post-processing is applied. Applying the *Filtering* operation (threshold = 0.5), improves the results substantially as it removes a vast number of false positive masks predicted by the model. Moreover, the *NMS* algorithm helps remove more of these masks as evident by the increase in *Food Picture Accuracy* by 10%. The *Join* improves the results slightly, with around 0.6% increase in *Food Picture Accuracy*.

To understand how sensitive the model is to the *Filtering threshold*, different values in the range of 0.1 - 0.9 are used while the *NMS* and *Join* steps are enabled. *Table* 5.5 summarizes these results. They show that as the *Filtering threshold* increases, the polygon precision increases since the model is more confident in the prediction, but the recall decreases substantially. This is the result of missing out on a lot of true positive masks that do not satisfy the threshold value. However, we can clearly see that the *Picture Accuracy* and *Food Picture Accuracy* are optimised around the 0.4 - 0.5 range.

ID	**Filtering threshold**	**Join threshold**	**NMS threshold**	**Piture Accuracy**	**Food Picture Accuracy**
V6	-	-	-	20.1	0.0
V7	0.5	-	-	35.4	19.9
V8	-	0.0	0.4	24.8	6.1
V9	0.5	0.0	-	36.1	20.8
V10	0.5	-	0.4	43.3	30.2
V11	0.5	0.0	0.4	**43.8**	**30.8**

TABLE 5.4: Analysis of the three post-processing steps used in the model.

Score threshold	**Picture Accuracy**	**Food Picture Accuracy**	**Ingredient Accuracy**	**Polygon Precision**	**Polygon Recall**
0.1	40.4	26.4	59.4	48.6	**68.9**
0.2	42.2	28.7	61.5	53.7	68.6
0.3	43.2	30.1	63.0	57.3	67.8
0.4	43.8	30.8	64.6	60.0	65.7
0.5	**43.8**	**30.8**	64.6	60.0	65.7
0.6	43.2	30.1	69.6	65.0	56.8
0.7	41.7	28.1	73.4	67.6	49.9
0.8	39.1	24.7	78.8	71.2	40.5
0.9	32.9	16.7	**87.2**	**78.2**	25.2

TABLE 5.5: Results of different filtering thresholds for the masks predicted by the model.

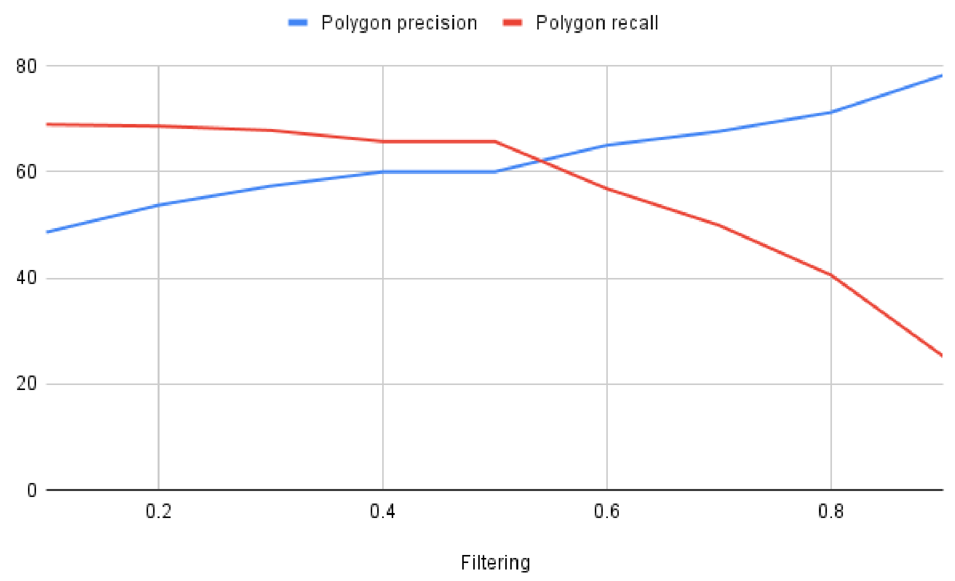


FIGURE 5.6: Polygon Precision and Recall against filtering thresholds in the range 0.1 - 0.9.

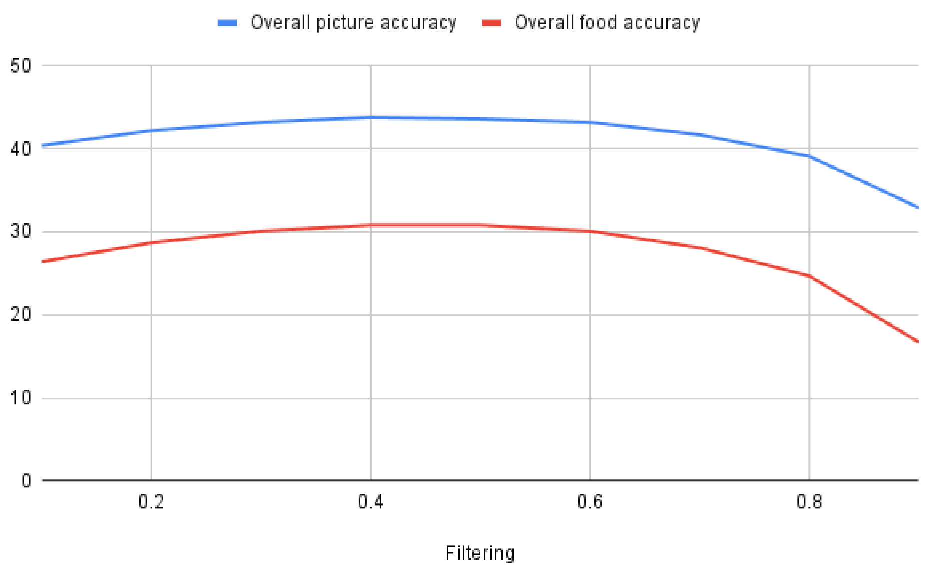


FIGURE 5.7: Picture Accuracy and Food Picture Accuracy against filtering thresholds in the range 0.1 - 0.9.

Chapter 6

Discussion

This thesis investigated the capabilities of computer vision models to detect food waste ingredients along with five image-level tasks in a supervised end-to-end framework. The focus of the study was to understand the effectiveness of multi-task learning on a large food waste dataset compiled by Orbisk. The outcome of this research can facilitate more efficient and accurate detection of food waste images, which would provide a deeper insight on the food waste problem. To tackle the six tasks presented in this study, a Mask2Former architecture [100] was altered to accommodate the other five tasks and trained in a supervised manner. Furthermore, the model is equipped with three post-processing algorithms that were not part of the original Mask2-Former model in order to further refine the masks and labels the model predicts. The challenge is to simplify the food waste processing pipeline by incorporating one model only without sacrificing the accuracy of predictions.

6.1 Findings and Interpretation

The main research question is *"Can a multi-task transformer framework be used to jointly handle segmentation, classification and regression tasks in food waste images without sacrificing accuracy?"*. The first step to answer the question is to establish a baseline for the performance of a comparable model on each task alone. Furthermore, the transformer framework was trained on subsets of tasks to validate its capability to handle multiple tasks together. Finally, the multi-task framework (FoodWasteAI) was placed in a test simulation using unseen data to analyse its performance as compared to Orbisk's current AI models.

The first two sub-questions were answered with the first two experiments and using the models' performance on the validation set. The final sub-question was addressed with the last test simulation where it competed with Orbisk's model on unseen data. In this section, the findings and results from this thesis is discussed along with an interpretation on how the models performed.

6.1.1 Multi-task Learning

One main focus of this thesis is to investigate the effectiveness of the multi-task learning paradigm on food waste images. This paradigm has been used successfully in a wide range of problems to incorporate similar tasks under the same model. For instance, several frameworks were proposed to handle scene detection [77], [78], tasks across different domains [89] and Food detection [99]. However, there is no in-depth research on combining segmentation, classification and regression tasks on food images.

To establish a baseline, single-task models were trained on each task separately and two variants of FoodWasteAI were trained on multiple tasks together. From *Table 5.1*, it can be seen that the single-task models performed slightly better on three classification tasks, namely container-type, has-waste and image-quality. On the other hand, the multi-task models outperformed the single-task ones on the remaining three tasks.

The has-waste task did not benefit from the multi-task learning paradigm, where it scored 0.2% higher accuracy with the single-task model as compared to the best FoodWasteAI variants. Similarly, the performance on the image-quality task deteriorated in multi-task models, with a decrease of 0.16% than its single-task counterpart. Most notably, the container-type task suffered the most reduction in performance in the multi-task variants, with a 0.68% decline as compared to the single-task model. On the other hand, the waste-stream task benefited from the combined training, which saw the FoodWasteAI perform with 0.17% increase in performance. The visual-weight saw a very margin increase in performance, with about 0.02 rise in the WMAPE metric. Most importantly, the FoodWasteAI model performed 0.1% better on the instance-segmentation task than its single-task baseline.

All in all, the difference in performance on each task between the FoodWasteAI and the single-task baselines is relatively negligible and can be attributed to random fluctuations in the model's parameters due to the random shuffling of the training data. However, it showcases the ability of the Swin backbone to handle multiple tasks in parallel and provide good representations for each task's head to decode a correct prediction. Furthermore, the FoodWasteAI-All model shows very similar performance on all classification and regression tasks as compared to the FoodWasteAI-Cls. This further proves that the ingredient-segmentation task does not have a substantial detriment on the other tasks when trained together.

6.1.2 FoodWasteAI

The FoodWasteAI architecture is inspired by the hard-parameter sharing proposed by Caruana (1993) [81]. The main advantage of such architecture is the simplicity and modularity of its parts. Most of the computations are done in a backbone model to arrive at a low-resolution feature vector that can be decoded by task heads to arrive at good predictions. This design showcased good results on all tasks mentioned in this study. The classification tasks have a strong performance with an accuracies in the range of 85% - 90%. The segmentation task also witnessed a a very strong mAP score of 27.6%. The current SOTA performance on the COCO instance segmentation dataset is 56%. However, the COCO dataset has 80 instance categories as compared to the 740 categories present in Orbisk's dataset.

Upon diving deeper in the ingredient-segmentation performance, the model seems to understand the different visual clues between the different ingredient labels and categories. From *Figure 5.4a*, it can be seen that the model performs substantially well on "Drinks", "Cheese", "Snacks", "Fruit", "Sandwiches" and "Vegetables". All these categories have a relatively distinct visual appearance which may explain this performance. On the other hand, the "Unknown" category shows the worst precision among all categories. This can be explained by the ambiguity of the category as it comprises any unknown ingredients, which may have very different visual appearance.

When analysing the precision in ingredient labels (*Figure 5.4b*), this ambiguity is further proven as the "unknownlabel" has the lowest precision among all labels. Other labels that suffer from low precision score are "sauceunknown_-brown_withvegetable", "chicken_fillet" and "noodles_short_withpieces". One possible reason for the low precision score is that these labels share the same category with other similar labels that have very margin visual discrepancy. For instance, there are multiple "sauce", "chicken" and "noodle" labels that describe different ways in which these ingredients are prepared. Conversely, the model performs well on "egg_cuttingwaste", "pizza_cuttingwaste", "bread-bun_croissant", "lemon_cut" and "cucumber_cut". Most of these labels are for chopped ingredients. This may explain their high precision score as they tent to have a distinct look.

6.1.3 Test Simulation

An important step in developing machine learning models is to evaluate them in real-world settings. This provides an understanding in the applicability of such models to address important applications such as food waste monitoring. The FoodWasteAI model is placed in a test simulation using unseen data to analyse its performance and compare it to Orbisk's current setup. The FoodWasteAI showed 3.1% increase on picture accuracy and 4.1% increase on food picture accuracy. This means that around one in three food images get all their ingredients correctly detected and classified.

These results are achieved with a remarkable reduction in model parameters and training time. Using single-task models to handle each task would result in having six models to train and deploy. This would account for 185.15M parameters to train and 326 hours of training time using the same GPU configurations used in this thesis. On the other hand, the FoodWasteAI model is able to achieve similar results on each task using only 47.69M parameters and a training time of 156 hours. This accounts for 75% reduction in parameters and 53% less training time. Moreover, the inference time is massively reduced as well since each image is passed through one model instead of six. When compared to Orbisk's current setup, the model outperforms them in the test simulation while using 35% less parameters and 25% less training time.

To summarize, the main research question regarding the applicability of ViT to handle multiple tasks together is answered positively. The FoodWasteAI model is applied successfully on six different tasks using the Orbisk food waste dataset. Although the performance can be further improved, the FoodWasteAI outperforms Orbisks's current models on a test simulation while being substantially more efficient to train.

6.1.4 Ablation Studies

The three post-processing algorithms used to refine the ingredient predictions, showcased a substantial improvement over the default Mask2Former decoding strategy. Although it is convenient to specify the maximum number of predictions per image, in some applications, such as food waste detection, it is unknown how many instances are visible on each image. The filtering stage allowed the model to filter out a lot of wrong predictions which improved the accuracy substantially. This step alone resulted in 15 - 20% increase in the business metrics. The NMS step also had a strong impact on the metrics with an improvement of 8 - 10%. The Join algorithm only had a slight effect on the accuracy but it still enhanced the results. During the qualitative analysis, the predictions obtained without these post-processing steps showed that for each ingredient, many masks were laid on top of each other with varying label values. Therefore merging or removing these masks was important in order to reduce the number of false-positive predictions made by the model.

The second ablation study investigated the effect of augmentations on the model's performance on each task. Given the limited time, each model variant was trained for a small number of iterations and evaluated on a subset of the validation set. The results show that only *Contrast* augmentation has a strong effect on the model's performance and helps it perform better. The *Brightness* augmentation also showed promising results but after combining it with the *Contrast* augmentation, it had a strong negative effect on the instance-segmentation task. Flipping and rotation augmentations produced bad results on the waste-stream, container-type and ingredient-segmentation tasks. One reason for this effect is the sensitivity of these tasks to the food location in the image. All in all, the final FoodWasteAI model is trained with *Contrast* augmentations only on the training set and uses the three post-processing algorithms to optimise the ingredient predictions.

6.2 Limitations

Although this study showcases promising results, they should be viewed alongside potential limitations. These limitations pertain to the model architecture used and the dataset collection which is done by Orbisk's devices.

6.2.1 FoodWasteAI

Although the model was designed with simplicity and modularity in mind, this choice might be detrimental to the its performance. The model only shares the backbone among the tasks. Bhattacharjee et al. (2022) showcased that an attention mechanism shared among the task heads improves the performance significantly [77]. In addition, Ye and Xu (2022) also showed that using the heads predictions to further refine the model's output results in better quality predictions. Moreover, the classification and regression heads used in FoodWasteAI are simple linear layers that decode the feature vector from the backbone. A more sophisticated head design can have a great effect on the model's performance on these tasks.

While training the models in this research, it was observed that the model does not converge properly at the end of the training cycle. Given the long training time each model requires and the limited hardware resources, the models could not be trained for longer. Given time and/or extra resources, the models should be trained longer in order for them to fully converge and fit the data. For reference, the Mask2Former model is trained for 50 epochs on the COCO dataset [106] which accounts for 368,750 iterations. This is three times longer than the training iterations of FoodWasteAI.

In this thesis, the backbone of choice was the Swin transformer due to its great results and applicability to various tasks. However, a deeper analysis of different backbones should be conducted to understand the effect of this component on the model's performance. Furthermore, larger models such as Swin-B and Swin-L were shown by to yield better results on classification and segmentation tasks [26], [61], [77], [78], [100]. However, they require even more time and resources to train.

The FoodWasteAI uses a simple linear combination of all tasks' losses in order to update the model's weights. Although this method is effective in training the model, it can result in significant drawbacks if the tasks' losses are not on the same scale. This will lead to one task dominating the other tasks which is undesired. There has been research in this area that aims help the model understand how to weigh each task. They range from fully learnable weights [91] to adaptive regularisation of gradient magnitudes [92]. Such methods elevate the need to find the optimum weights for each task which takes a long time depending on the speed of training. Given the limited time of this research and the long training time of the model's, only linear combination is used to train the model.

6.2.2 Orbisk Dataset

Although the theoretical findings of this research is generic, they are influenced by the conditions of Orbisk's data collection. Therefore, the generalizability of these results should be considered when interpreting the results. The Orbisk food waste dataset is a collection of images taken by their hardware devices placed in catering and hospitality institutes, mainly in the Netherlands. Consequently, this places a bias on the type of food waste images taken as it tends to be Dutch food.

Given that the Orbi is a camera put on top of the waste bin, most images have the food waste somewhere in the middle of the frame with the bin and floor in the background. As a result, a model trained on these images may struggle to deal with food waste images taken in a different environment.

The Orbisk's dataset is manually annotated by their external annotators that are instructed to annotate each image using a predefined guideline on how they should annotate. This guideline is designed to optimise Orbisk's business process and may not provide the perfect annotations for each image. In addition, given the nature of how the images are taken, some of them suffer from technical imperfections such as motion blur, occlusions and inconsistent lighting conditions.

6.3 Future Direction

The findings and limitations provide interesting paths for future research. Firstly, the model architecture can be altered to accommodate more sophisticated method of combining the tasks. Implementing a shared attention mechanism between tasks in order to allow each task head to influence the other heads. This may lead to promising results as shown by [77], [78], [99]. Furthermore, identifying the optimal backbone to use would provide richer features to the task heads.

Another promising direction is using a multi-modality approach. One such cases is the *Segment anything* model which uses a variety of prompts to process and segment the image [113]. The model allows prompts to be fed alongside the image and the model process them to provide an output. OneFormer is another framework that uses multi-modal input to process an image [114].

The model is trained once and can be used for any instance, semantic or panoptic segmentation tasks by providing a task token with the image. A similar approach can be used with FoodWasteAI in order to specify which tasks the model should perform on the image.

Finally, one future research can investigate the effect of iterative fine-tuning on continuous data that arrives periodically. For instance, Orbisk receives new images every day from their clients and their models are retrained periodically to accommodate these new images. An interesting study can research different techniques to optimise this re-training step in order to improve performance and reduce training time.

Chapter 7

Conclusion

This thesis presented a multi-task learning framework trained to handle six different tasks concurrently. The motivation behind this work is to provide an understanding of the effectiveness of multi-task learning to process food waste images. The model is able to outperform the current AI models at Orbisk with substantially less parameters and training time. In addition, the FoodWasteAI performs better than the single-task counterparts on three out of the six tasks presented in this study. All in all, the FoodWasteAI is ready to replace Orbisk's current models, which would boost their AI accuracy and efficiency.

Moving forward, the limitations of this research should be addressed to build upon the work presented. Incorporating a sophisticated mechanism to fuse the heads together would allow each task head to integrate the other tasks' output in its prediction. Furthermore, improving the automated detection of food waste would provide an important insight on how to effectively reduce wasting food.

Bibliography

[1] U. N. D. of Economic and S. Affairs, *The Sustainable Development Goals: Report 2022*. UN, 2022.

[2] Z. Zhongming, L. Linong, Y. Xiaona, L. Wei, *et al.*, "Unep food waste index report 2021," 2021.

[3] B. Lipinski, C. Hanson, R. Waite, T. Searchinger, and J. Lomax, "Reducing food loss and waste," 2013.

[4] A. Pandey, "Food Wastage: Causes, Impacts And Solutions," *Science Heritage Journal (GWS)*, vol. 5, no. 1, pp. 17–20, 2021.

[5] WHO, FAO, IFAD, UNICEF, and WFP, "The state of food security and nutrition in the world 2022," Rome, Italy, Tech. Rep., 2022, Pages: #260 p.

[6] "Monitoring food loss and waste essential to hunger fight." (), [Online]. Available: https://www.fao.org/news/story/en/item/203149/icode/.

[7] C. Mbow, C. Rosenzweig, L. G. Barioni, *et al.*, "Food security," 2019.

[8] "Deforestation and forest degradation." (), [Online]. Available: https://www.worldwildlife.org/threats/deforestation-and-forest-degradation.

[9] "Food waste and food waste prevention - estimates." (2022), [Online]. Available: https://ec.europa.eu/eurostat/statistics-explained/index.php?.

[10] B. Kretschmer, C. Smith, E. Watkins, *et al.*, "Recycling agricultural, forestry & food wastes and residues for sustainable bioenergy and biomaterials (part of the project'technology options for feeding 10 billion people')," 2013.

[11] RaboBank and N. Cappetti. "Rabobank: Forse vermindering voedselverspilling in horeca mogelijk." (2020), [Online]. Available: https://www.rabobank.com/nl/press/search/2020/20200113-rabobank-forse-vermindering-voedselverspilling-in-horeca-mogelijk.html.

[12] R. Jagt. "Food waste has gone viral." (2020), [Online]. Available: https://www2.deloitte.com/nl/nl/pages/consumer/articles/food-covid-19-food-waste-gone-viral.html.

[13] D. Leverenz, G. Hafner, S. Moussawel, M. Kranert, Y. Goossens, and T. Schmidt, "Reducing food waste in hotel kitchens based on self-reported data," *Industrial Marketing Management*, vol. 93, pp. 617–627, 2021.

[14] "Reduce your food waste." (), [Online]. Available: https://orbisk.com/en.html.

[15] H. Zhang, C. Wu, Z. Zhang, *et al.*, "Resnest: Split-attention networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2022, pp. 2736–2746.

[16] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[17] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.

[18] J. Baxter, "A model of inductive bias learning," *Journal of artificial intelligence research*, vol. 12, pp. 149–198, 2000.

[19] W. Min, L. Liu, Z. Wang, *et al.*, "Isia food-500: A dataset for large-scale food recognition via stacked global-local attention network," in *Proceedings of the 28th ACM International Conference on Multimedia*, ser. MM '20, Seattle, WA, USA: Association for Computing Machinery, 2020, 393–401.

[20] G. M. Farinella, D. Allegra, M. Moltisanti, F. Stanco, and S. Battiato, "Retrieval and classification of food images," *Computers in Biology and Medicine*, vol. 77, pp. 23–39, 2016.

[21] F. Zhu, M. Bosch, C. J. Boushey, and E. J. Delp, "An image analysis system for dietary assessment and evaluation," in *2010 IEEE International Conference on Image Processing*, 2010, pp. 1853–1856.

[22] Y. He, C. Xu, N. Khanna, C. J. Boushey, and E. J. Delp, "Analysis of food images: Features and classification," in *2014 IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 2744–2748.

[23] L. Kabbai, M. Abdellaoui, and A. Douik, "Image classification by combining local and global features," *The Visual Computer*, vol. 35, no. 5, pp. 679–693, 2019.

[24] D. Linsley, J. Kim, V. Veerabadran, C. Windolf, and T. Serre, "Learning long-range spatial dependencies with horizontal gated recurrent

units," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018.

[25] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[26] Z. Liu, H. Hu, Y. Lin, *et al.*, "Swin transformer v2: Scaling up capacity and resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 12 009–12 019.

[27] A. Hatamizadeh, H. Yin, J. Kautz, and P. Molchanov, "Global context vision transformers," *arXiv preprint arXiv:2206.09959*, 2022.

[28] Q. Zhou, H. Zou, and H. Wu, "Lgvit: A local and global vision transformer with dynamic contextual position bias using overlapping windows," *Applied Sciences*, vol. 13, no. 3, p. 1993, 2023.

[29] K. Patel, A. M. Bur, F. Li, and G. Wang, "Aggregating global features into local vision transformer," in *2022 26th International Conference on Pattern Recognition (ICPR)*, 2022, pp. 1141–1147.

[30] P. Sundt, *Prevention of food waste in restaurants, hotels, canteens and catering*. Nordic Council of Ministers, 2012.

[31] K. Silvennoinen, L. Heikkilä, J.-M. Katajajuuri, and A. Reinikainen, "Food waste volume and origin: Case studies in the finnish food service sector," *Waste Management*, vol. 46, pp. 140–145, 2015.

[32] J. Bharucha, "Tackling the challenges of reducing and managing food waste in mumbai restaurants," *British Food Journal*, vol. 120, no. 3, pp. 639–649, 2018.

[33] V. Filimonau, H. Zhang, and L. en Wang, "Food waste management in shanghai full-service restaurants: A senior managers' perspective," *Journal of Cleaner Production*, vol. 258, p. 120 975, 2020.

[34] M. Eriksson, C. Malefors, P. Callewaert, H. Hartikainen, O. Pietiläinen, and I. Strid, "What gets measured gets managed – or does it? connection between food waste quantification and food waste reduction in the hospitality sector," *Resources, Conservation & Recycling: X*, vol. 4, p. 100 021, 2019.

[35] V. Filimonau, V. N. Nghiem, and L. en Wang, "Food waste management in ethnic food restaurants," *International Journal of Hospitality Management*, vol. 92, p. 102 731, 2021.

[36] K. Silvennoinen, S. Nisonen, and O. Pietiläinen, "Food waste case study and monitoring developing in finnish food services," *Waste Management*, vol. 97, pp. 97–104, 2019.

[37] V. Filimonau and V. A. Ermolaev, "A sleeping giant? food waste in the foodservice sector of russia," *Journal of Cleaner Production*, vol. 297, p. 126 705, 2021.

[38] V. Filimonau and J. Sulyok, "'bin it and forget it!': The challenges of food waste management in restaurants of a mid-sized hungarian city," *Tourism Management Perspectives*, vol. 37, p. 100 759, 2021.

[39] C. Martin-Rios, A. Hofmann, and N. Mackenzie, "Sustainability-oriented innovations in food waste management technology," *Sustainability*, vol. 13, no. 1, p. 210, 2020.

[40] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[41] K. Fukushima, "Cognitron: A self-organizing multilayered neural network," *Biological cybernetics*, vol. 20, no. 3-4, pp. 121–136, 1975.

[42] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, 2022.

[43] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[44] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[46] A. Cauchy *et al.*, "Méthode générale pour la résolution des systemes d'équations simultanées," *Comp. Rend. Sci. Paris*, vol. 25, no. 1847, pp. 536–538, 1847.

[47] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[48] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization.," *Journal of machine learning research*, vol. 12, no. 7, 2011.

[49] S. Bozinovski and A. Fulgosi, "The influence of pattern similarity and transfer learning upon training of a base perceptron b2," in *Proceedings of Symposium Informatica*, vol. 3, 1976, pp. 121–126.

[50] I. Alhashim and P. Wonka, "High quality monocular depth estimation via transfer learning," *arXiv preprint arXiv:1812.11941*, 2018.

[51] C. Raffel, N. Shazeer, A. Roberts, *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, 84–90, 2017.

[53] H. Kagaya, K. Aizawa, and M. Ogawa, "Food detection and recognition using convolutional neural network," in *Proceedings of the 22nd ACM International Conference on Multimedia*, ser. MM '14, Orlando, Florida, USA: Association for Computing Machinery, 2014, 1085–1088.

[54] S. Mezgec and B. Koroušić Seljak, "NutriNet: A deep learning food and drink image recognition system for dietary assessment," en, *Nutrients*, vol. 9, no. 7, Jun. 2017.

[55] P. Pouladzadeh and S. Shirmohammadi, "Mobile multi-food recognition using deep learning," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 13, no. 3s, 2017.

[56] D. Sahoo, W. Hao, S. Ke, *et al.*, "Foodai: Food image recognition via deep learning for smart food logging," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19, Anchorage, AK, USA: Association for Computing Machinery, 2019, 2260–2268.

[57] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[58] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[59] Y. Zhu, X. Zhao, C. Zhao, J. Wang, and H. Lu, "Food det: Detecting foods in refrigerator with supervised transformer network," *Neurocomputing*, vol. 379, pp. 162–171, 2020.

[60] J. Song, W. Min, Y. Liu, Z. Li, S. Jiang, and Y. Rui, "A noise-robust locality transformer for fine-grained food image retrieval," in *2022 IEEE*

5th International Conference on Multimedia Information Processing and Retrieval (MIPR), 2022, pp. 348–353.

[61] Z. Liu, Y. Lin, Y. Cao, *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 10 012–10 022.

[62] G. Sheng, S. Sun, C. Liu, and Y. Yang, "Food recognition via an efficient neural network with transformer grouping," *International Journal of Intelligent Systems*, vol. 37, no. 12, pp. 11 465–11 481, 2022.

[63] X. Wu, X. Fu, Y. Liu, E.-P. Lim, S. C. Hoi, and Q. Sun, "A large-scale benchmark for food image segmentation," in *Proceedings of the 29th ACM International Conference on Multimedia*, ser. MM '21, Virtual Event, China: Association for Computing Machinery, 2021, 506–515.

[64] Y. Matsuda, H. Hoashi, and K. Yanai, "Recognition of multiple-food images by detecting candidate regions," in *Proc. of IEEE International Conference on Multimedia and Expo (ICME)*, 2012.

[65] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101 – mining discriminative components with random forests," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, pp. 446–461.

[66] W. Min, Z. Wang, Y. Liu, *et al.*, "Large scale visual food recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–18, 2023.

[67] A. Salvador, N. Hynes, Y. Aytar, *et al.*, "Learning cross-modal embeddings for cooking recipes and food images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[68] G. Ciocca, P. Napoletano, and R. Schettini, "Food recognition and leftover estimation for daily diet monitoring," in *New Trends in Image Analysis and Processing – ICIAP 2015 Workshops*, V. Murino, E. Puppo, D. Sona, M. Cristani, and C. Sansone, Eds., Cham: Springer International Publishing, 2015, pp. 334–341.

[69] G. Ciocca, P. Napoletano, and R. Schettini, "Food recognition: A new dataset, experiments and results," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 3, pp. 588–598, 2017.

[70] T. Ege, W. Shimoda, and K. Yanai, "A new large-scale food image segmentation dataset and its application to food calorie estimation based on grains of rice," in *Proceedings of the 5th International Workshop*

on Multimedia Assisted Dietary Management, ser. MADiMa '19, Nice, France: Association for Computing Machinery, 2019, 82–87.

[71] K. Okamoto and K. Yanai, "Uec-foodpix complete: A large-scale food image segmentation dataset," in *Pattern Recognition. ICPR International Workshops and Challenges*, A. Del Bimbo, R. Cucchiara, S. Sclaroff, *et al.*, Eds., Cham: Springer International Publishing, 2021, pp. 647–659.

[72] M. Bosch, F. Zhu, N. Khanna, C. J. Boushey, and E. J. Delp, "Combining global and local features for food identification in dietary assessment," in *2011 18th IEEE International Conference on Image Processing*, 2011, pp. 1789–1792.

[73] V. Bettadapura, E. Thomaz, A. Parnami, G. D. Abowd, and I. Essa, "Leveraging context to support automated food recognition in restaurants," in *2015 IEEE Winter Conference on Applications of Computer Vision*, 2015, pp. 580–587.

[74] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, 41–75, 1997.

[75] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08, Helsinki, Finland: Association for Computing Machinery, 2008, 160–167.

[76] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015.

[77] D. Bhattacharjee, T. Zhang, S. Süsstrunk, and M. Salzmann, "Mult: An end-to-end multitask learning transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 12 031–12 041.

[78] H. Ye and D. Xu, "Inverted pyramid multi-task transformer for dense scene understanding," in *Computer Vision – ECCV 2022*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds., Cham: Springer Nature Switzerland, 2022, pp. 514–530.

[79] Y. Yang and T. M. Hospedales, "Trace norm regularised deep multi-task learning," *arXiv preprint arXiv:1606.04038*, 2016.

[80] L. Duong, T. Cohn, S. Bird, and P. Cook, "Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser,"

in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 845–850.

[81] R. Caruana, "Multitask learning: A knowledge-based source of inductive bias," in *ICML*, 1993.

[82] M. Long, Z. CAO, J. Wang, and P. S. Yu, "Learning multiple tasks with multilinear relationship networks," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[83] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-stitch networks for multi-task learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[84] Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi, and R. Feris, "Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[85] M. Teichmann, M. Weber, M. Zöllner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1013–1020.

[86] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[87] I. Kokkinos, "Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[88] Y. Liao, S. Kodagoda, Y. Wang, L. Shi, and Y. Liu, "Understand scene categories by objects: A semantic regularized scene classifier using convolutional neural networks," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 2318–2325.

[89] R. Hu and A. Singh, "Unit: Multimodal multitask learning with a unified transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 1439–1449.

[90] T. Standley, A. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese, "Which tasks should be learned together in multi-task learning?" In *Proceedings of the 37th International Conference on Machine Learning*, H. D.

III and A. Singh, Eds., ser. Proceedings of Machine Learning Research, vol. 119, PMLR, 2020, pp. 9120–9132.

[91] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[92] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 794–803.

[93] P. Liu, X. Qiu, and X. Huang, "Adversarial multi-task learning for text classification," *arXiv preprint arXiv:1704.05742*, 2017.

[94] O. Sener and V. Koltun, *Multi-task learning as multi-objective optimization*, 2018.

[95] X. Lin, H.-L. Zhen, Z. Li, Q. Zhang, and S. Kwong, *Pareto multi-task learning*, 2019.

[96] X. Lin, Z. Yang, Q. Zhang, and S. Kwong, *Controllable pareto multi-task learning*, 2020.

[97] X.-J. Zhang, Y.-F. Lu, and S.-H. Zhang, "Multi-task learning for food identification and analysis with deep convolutional neural networks," *Journal of Computer Science and Technology*, vol. 31, no. 3, pp. 489–500, 2016.

[98] T. Ege and K. Yanai, "Multi-task learning of dish detection and calorie estimation," in *Proceedings of the Joint Workshop on Multimedia for Cooking and Eating Activities and Multimedia Assisted Dietary Management*, ser. CEA/MADiMa '18, Stockholm, Sweden: Association for Computing Machinery, 2018, 53–58.

[99] H. Liang, G. Wen, Y. Hu, M. Luo, P. Yang, and Y. Xu, "Mvanet: Multitask guided multi-view attention network for chinese food recognition," *IEEE Transactions on Multimedia*, vol. 23, pp. 3551–3561, 2021.

[100] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention mask transformer for universal image segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 1290–1299.

[101] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*, Springer, 2020, pp. 213–229.

[102] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[103] B. Cheng, A. Schwing, and A. Kirillov, "Per-pixel classification is not all you need for semantic segmentation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 864–17 875, 2021.

[104] W. Zhang, J. Pang, K. Chen, and C. C. Loy, "K-net: Towards unified image segmentation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 10 326–10 338, 2021.

[105] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[106] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, Springer, 2014, pp. 740–755.

[107] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: An overview," *arXiv preprint arXiv:2008.05756*, 2020.

[108] K. Chen, J. Wang, J. Pang, *et al.*, "MMDetection: Open mmlab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019.

[109] M. Contributors, *Openmmlab's pre-training toolbox and benchmark*, `https://github.com/open-mmlab/mmpretrain`, 2023.

[110] *Mlflow*, `https://mlflow.org/`.

[111] B. E. Moore and J. J. Corso, "Fiftyone," *GitHub. Note: `https://github.com/voxel51/fiftyone`*, 2020.

[112] O. Russakovsky, J. Deng, H. Su, *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[113] A. Kirillov, E. Mintun, N. Ravi, *et al.*, "Segment anything," *arXiv:2304.02643*, 2023.

[114] J. Jain, J. Li, M. Chiu, A. Hassani, N. Orlov, and H. Shi, "OneFormer: One Transformer to Rule Universal Image Segmentation," 2023.

Appendix A

Losses and Activation Functions Equations

Cross-entropy Loss

$$\mathcal{L}_{ce} = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{A.1}$$

where M is the number of classes, y is a binary indicator whether the label c is the correct classification label for observation o and p is the predicted probability of observation o is of class c.

Smooth L1 Loss

$$L_\delta = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & \text{if } |(y-\hat{y})| < \delta \\ \delta((y-\hat{y}) - \frac{1}{2}\delta) & \text{otherwise} \end{cases} \tag{A.2}$$

where y is the predicted values, $\hat{y}$ is the ground-truth value and δ = 2.0

ReLU

$$f(x) = max(0, x) = \begin{cases} x_i & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{A.3}$$

Sigmoid

$$f(x) = \frac{1}{1 + e^{-x+}} \tag{A.4}$$

Softmax

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{A.5}$$

Appendix B

Training History

Single-Task Models

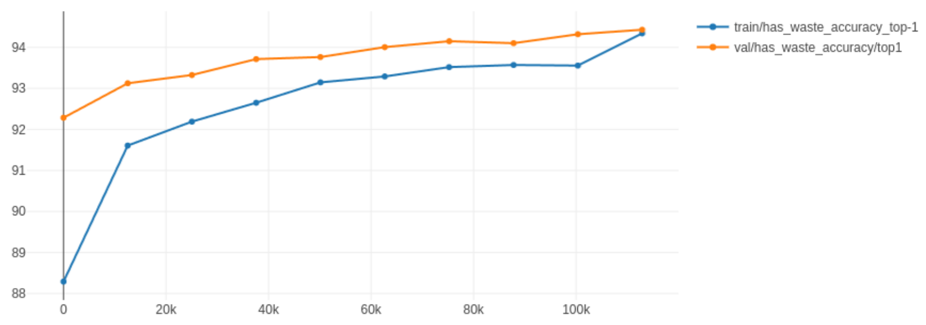


FIGURE B.1: The performance history on the has-waste task during training.

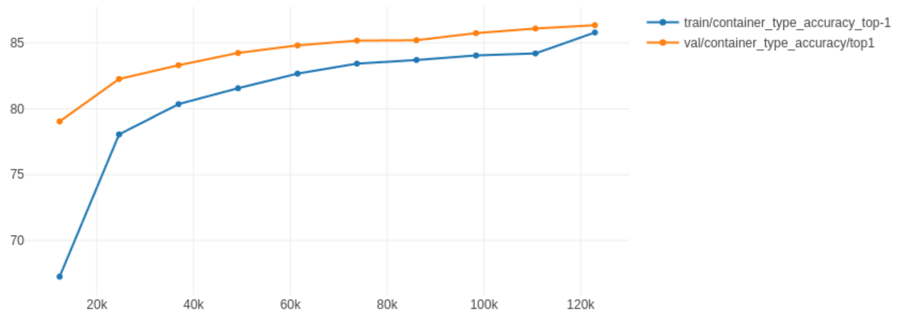


FIGURE B.2: The performance history on the container-type task during training.

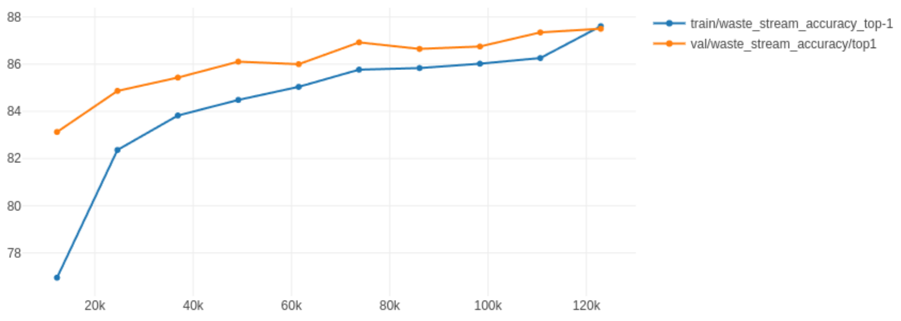


FIGURE B.3: The performance history on the waste-stream task during training.

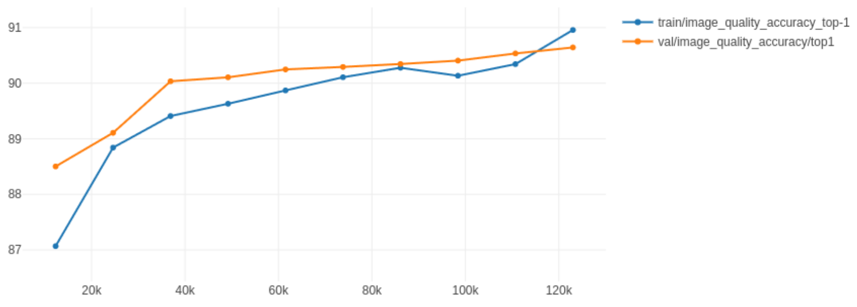


FIGURE B.4: The performance history on the image-quality task during training.

FoodWasteAI

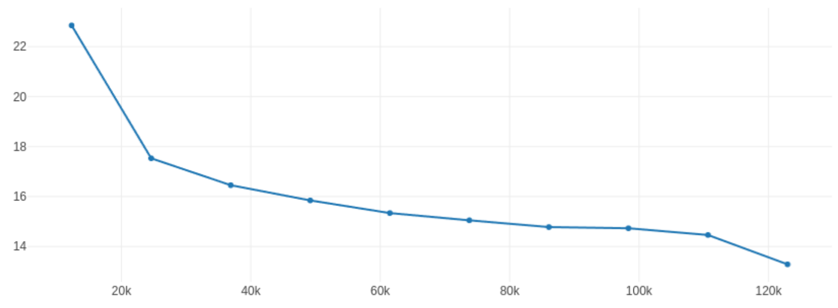


FIGURE B.5: History of total loss during training.

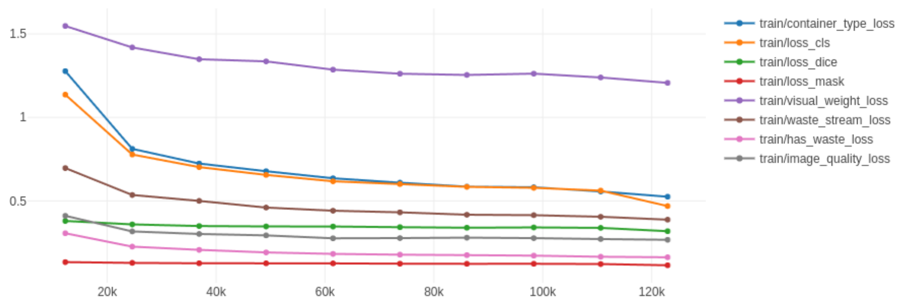


FIGURE B.6: History of task losses during training. Note that, "loss_cls", "loss_dice" and "loss_mask" are all used to train the segmentation head.

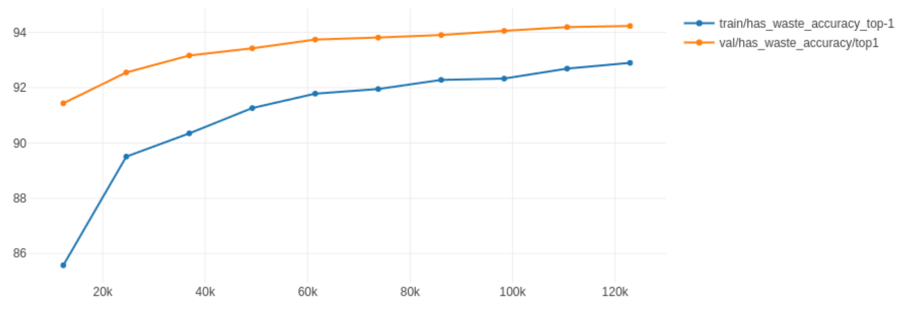


FIGURE B.7: History of has-waste task during training.

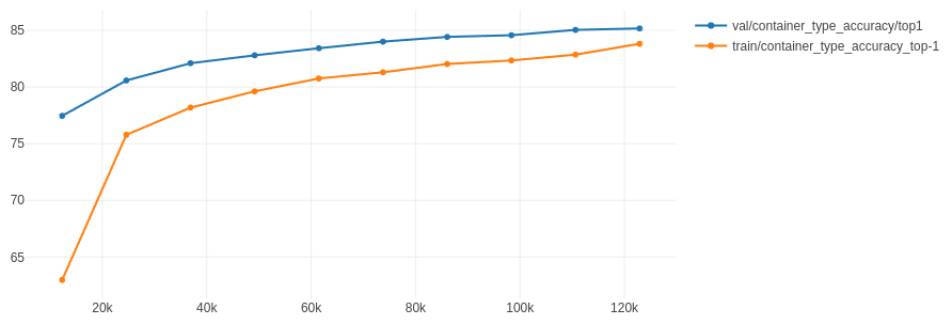


FIGURE B.8: History of container-type task during training.

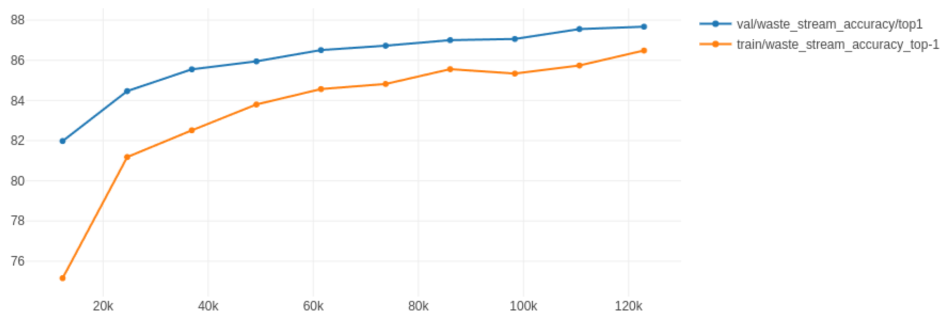


FIGURE B.9: History of waste-stream task during training.

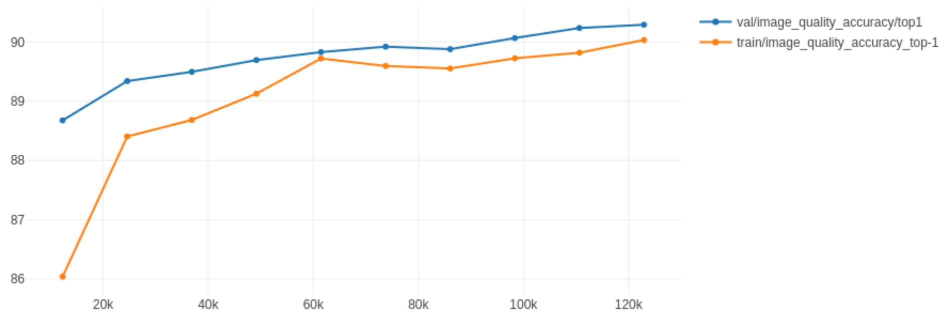


FIGURE B.10: History of image-quality task during training.

Appendix C

Task Labels and Augmentations

Waste-stream label	**Image-quality label**
Unused ingredient	Good
Prepared food	Blurry
Peels	Overexposed
Served food	Underexposed
Mixed	Camera issue
None	None

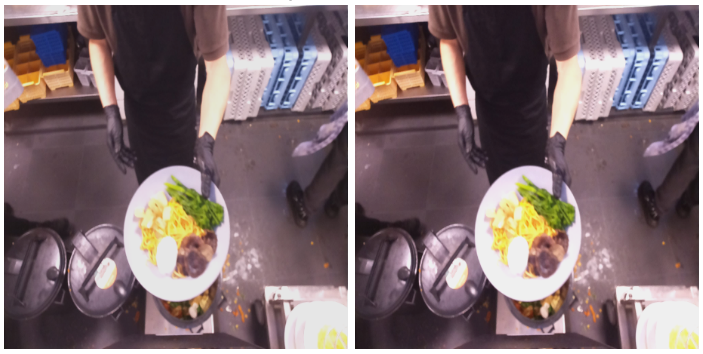
TABLE C.1: The waste-stream and image-quality labels used at Orbisk.

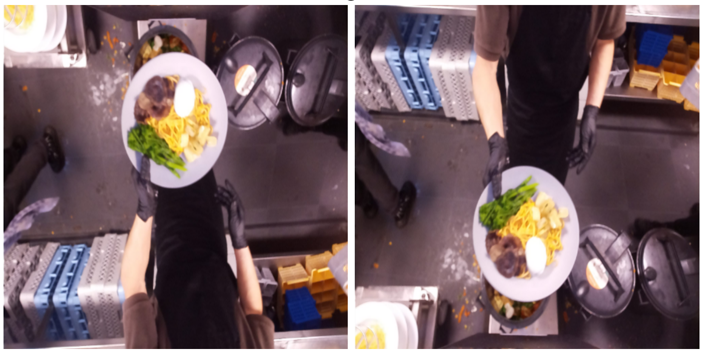
Container-type label

Bottle	Container metal gastronorm	Crate
Bowl ceramic	Container metal square	Dishwasher tray
Bowl glass	Container plastic	Dustpan
Bowl metal	Cup glass	Multiple
Bread basket	Cutting board	No container
Carafe	Cup mug	Original packaging
Other	Tray metal	Pan
Plate	Tray plastic	bin
Sieve	Tray wood	Tray ceramic
Tray cardboard	None	

TABLE C.2: The container-type labels used at Orbisk.

(A) Original and Contrast


(B) Brightness


(C) Flip

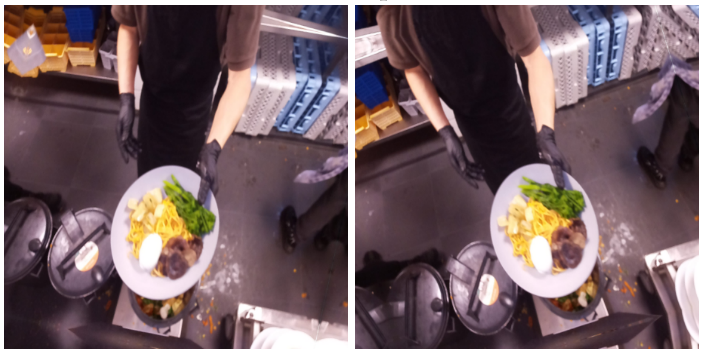

(D) Rotation

FIGURE C.1: Augmentations used in the ablation study. a shows the original image on the right and the contrast on the left. The contrast factor is chosen uniformly from 0.8 - 1.2. b brightness factor is chosen uniformly from 0.8 - 1.2. c random flipping horizontally, vertically or both. d random rotation chosen uniformly from -45° - 45°.

Appendix D

Preliminary Results

These preliminary results were conducted as a research internship at Orbisk prior to starting my thesis. These results show the positive impact of multi-task learning on the classification tasks in the Orbisk's dataset. The best multi-task variants produced better performance per task than the single-task ones. This shows that the classification tasks benefit from the combined training.

	Weights			Accuracy			
ID	**Waste Stream**	**Container Type**	**Has Waste**	**Waste Stream**	**Container Type**	**Has Waste**	**Combined**
1	0.333	0.333	0.333	88.0	87.6	**95.0**	81.7
2	0.212	0.212	0.516	87.8	87.5	**95.0**	81.5
3	0.107	0.107	0.787	87.4	86.9	**95.0**	80.8
4	0.212	0.576	0.212	87.8	87.8	94.8	81.8
5	0.155	0.422	0.422	87.8	87.7	94.9	81.8
6	0.090	0.245	0.665	87.5	87.5	**95.0**	81.1
7	0.107	0.787	0.107	87.6	**88.0**	**95.0**	81.5
8	0.090	0.665	0.245	87.6	**88.0**	94.9	81.7
9	0.063	0.468	0.468	87.6	87.8	**95.0**	81.5
10	0.576	0.212	0.212	**88.1**	87.4	94.8	81.7

TABLE D.1: Performance of the first 15 models trained in the preliminary study. Best performance in each task is highlighted in bold font. "Combined" specifies the percentage of images that have all tasks correctly predicted.

	Weights			Accuracy			
ID	**Waste Stream**	**Container Type**	**Has Waste**	**Waste Stream**	**Container Type**	**Has Waste**	**Combined**
11	0.422	0.155	0.422	**88.1**	87.3	94.9	81.6
12	0.245	0.090	0.665	87.6	86.8	94.9	80.8
13	0.422	0.422	0.155	**88.1**	87.8	**95.0**	81.9
14	0.245	0.665	0.090	87.7	87.9	94.7	81.9
15	0.787	0.107	0.107	87.9	86.8	94.9	81.0
16	0.665	0.090	0.245	87.9	86.7	94.9	80.9
17	0.468	0.063	0.468	87.8	86.6	94.9	80.9
18	0.665	0.245	0.090	87.8	87.3	94.7	81.5
19	0.468	0.468	0.063	**88.1**	**88.0**	94.8	**82.2**
20	1	0	0	87.6	2.5	43.3	0.7
21	0	1	0	16.3	**88.0**	55.1	1.6
22	0	0	1	0.4	5.4	94.7	0.0
23	auto	auto	auto	88.0	87.5	94.9	81.7

TABLE D.2: Performance of remaining models trained in the preliminary study. Best performance in each task is highlighted in bold font. "Combined" specifies the percentage of images that have all tasks correctly predicted. "auto" means that the automatic weight objective function in Equation 2.3 is used.