ACIT5900

MASTER THESIS

in

Applied Computer and Information Technology (ACIT)

May 2024

Applied Artificial Intelligence

Can XAI-guided Point-of-Interest Proposal and SAM Perform Crossmodal Food Image Segmentation?

Rikard Haukemyr Donnelly

Department of Computer Science

Faculty of Technology, Art and Design

OSLOMET

Preface

In this master thesis, I aim to use methods from explainable AI to add additional modalities to existing models. As AI can be both expensive and time-consuming to train, my hope is that leveraging a weak form of object localization can provide a text prompting interface for a stronger image segmentation model without text input options. This journey proved challenging, as I ran into several problems, but in the end I believe my thesis provides an interesting perspective.

Choosing a thesis topic can be difficult - you never know where it will lead you. When my supervisor suggested working on food image segmentation with explainable AI integration, I knew I had chosen the right topic for me. As far as my interests go, image processing is a topic close to my heart. Additionally, I see interpretability and alignment as relevant topics for AI, both as interesting research topics and for their importance to society.

I would very much like to thank Raju Shrestha and Ayan Chatterjee for their immense support and guidance as my supervisors this semester. I would also like to thank my family for how they raised me to be curious about the world. Last but not least, I'd like to thank all my friends, new and old, who have kept my spirits high while studying. All of you have made my journey worthwhile.

Oslo, 2024-05-15 Rikard H. Donnelly

Abstract

Food image segmentation is a task with multiple problems. There are limited food image segmentation datasets, considerable variability in the types of food available across geographical regions and ethnicities, and adapting to new modalities and domains can be resource intensive.

In this thesis, we explore the use of methods from explainable AI as a way to give new modalities general-purpose segmentation models. It achieves this by using CLIP as a multimodal model with a shared embedding space for text and image data. We propose a system with modular components that integrate to form a promptable text to image segmentation model. Through an extensive set of hyperparameter impact studies, we evaluate how the system performs under different configurations, and determine which components have the highest impact on the system's performance.

We found that using techniques from explainable AI together with filtering techniques and the sampling of points of interest to be highly effective in guiding a general-purpose segmentation model. While we did not beat state of the art models, our results were reasonably competitive. However, we had problems developing a classifier for dishes and ingredients using high-dimensional embeddings and k-nearest neighbors search.

In conclusion, the research proved interesting, and deserves further research. We were able to segment food images under supervision. However, our classification results did not prove fruitful in this thesis. We conjecture that using a stronger multimodal model like SigLiT could improve our results. Additionally, both the classification and segmentation tasks could be improved by fine-tuning the respective component on a large food-specific dataset like Recipe1M.

Table of Contents

List of Figures

List of Tables

1 - Introduction

1.1 - Introduction

All humans consume food and drinks in one way or another. This reliance on nutrition brings with it multiple challenges, like how to properly balance your diet (*Helsedirektoratets kostråd*, 2019) and the need for measures to ensure public safety (Meenu et al., 2021).

To assist with food-related challenges, methods from computer vision have been applied to a variety of tasks. For instance, computer vision can help with food and ingredient recognition (Wu et al., 2021), quality and safety assessment (Khan et al., 2021; Meenu et al., 2021), portion size estimation (Konstantakopoulos et al., 2024; Lo et al., 2020), calorie counting (Meyers et al., 2015), and recipe generation (Chhikara et al., 2023; Yin et al., 2023).

Several food-related tasks use image segmentation to know what parts of the image contain the types of food we are interested in. This task aims to identify regions of an image that contain the relevant food items. If a human were to do this, they might use a transparent sheet of plastic and highlighters of different colors to show what parts of the image contained what type of food. For calorie counting, you might highlight each pixel of a hamburger, estimate its area and volume, convert the volume to an estimated weight, then multiply the weight by calories per weight unit. This gives you an estimated calorie count for the hamburger. All of this relies on segmenting the image to determine where the hamburger is and how much space it takes up.

Despite these advancements, significant challenges remain in developing models that are both accurate and efficient in segmenting food images. This is where the integration of advanced AI techniques comes into play.

1.2 - Motivation

A drawback of many existing AI models is that making them from scratch requires tremendous amounts of training data and considerable computation time. General purpose models like CLIP (Contrastive Language-Image Pre-training), SAM (Segment Anything Model) and DINO (Self-distillation With No Labels) are available as pretrained alternatives that can be adapted through fine-tuning (Caron et al., 2021; Kirillov et al., 2023; Radford et al., 2021; X. Wang et al., 2023). This reduces the burden of those who want to implement machine learning systems.

However, researchers do not always release their models. In the case of SAM, Meta did not release the LLM-augmented segmentation models they researched (Kirillov et al., 2023). At the time of release, DINO was a good general purpose segmenter, but it did not possess prompting abilities.

Science is a process that intends to improve over time. (Huang et al., 2024) explored

various methods for prompting SAM with text and all the modalities. (S. Liu et al., 2023) used grounded pre-training to make DINO multi-modal with text-promptable segmentation. These are but a few examples of how models can be adapted to new tasks and domains through collaboration and innovative research. However, (Huang et al., 2024; S. Liu et al., 2023) still required explicit retraining to achieve their results.

Thus, some questions arise: if we have limited data and computational resources, how might we still be able to perform guided food image segmentation? Can we leverage some of these strengths of the methods mentioned before? Also, how far can we get with general-purpose models?

We might be able to combine some of the models above, like CLIP and SAM. After all, (Kirillov et al., 2023) already tried using CLIP as the language embedder, but did not release their text prompt encoder. It is difficult to see how well this worked for them, because they didn't release quantitative results, saying that “Our foray into the text-to-mask task is exploratory and not entirely robust, although we believe it can be improved with more effort.” The problem would then be finding a way to combine the segmentation properties of SAM with the multimodal properties of CLIP in a way that minimizes the computational resources required.

Our idea for bridging this gap involves using XAI (Explainable AI) techniques. XAI can generate masks from multimodal text-and-image models like CLIP (Choe et al., 2020; Petryk et al., 2022; D. Zhang et al., 2022). The masks generated by explainable AI correlate with, but might not completely match ground-truth segmentation masks (Selvaraju et al., 2020). However, they might be good enough to guide a segmentation model like SAM.

This approach has both benefits and drawbacks. One benefit is that the method doesn't necessarily need retraining or additional data. This hinges on the general purpose models being able to capture enough semantic context about the data, which is also the most prevalent drawback we foresee. We would be able to generate segmentations from this method, but we would also need to know what segmentation maps to attempt to generate.

1.3 - Problem Statement

There are many different explainability techniques available. The explainability techniques we plan to use fall into a category called CAM (Class Activation Maps) (Selvaraju et al., 2020). Using CAM requires the text label to be predicted in advance. We will likely need a classifier to identify the expected classes in an image to segment them. (Radford et al., 2021) shows that nearest-neighbor retrieval is possible, and other lightweight machine learning techniques are available. The question is whether they perform well enough for food image classification.

Since the proposed technique relies on explainability techniques to generate segmentation maps, we need to know if those segmentation maps are useful.

The segmentation maps might contain noise. Small non-zero probabilities can add up and potentially lead the segmentation model astray. We need to explore whether we can improve XAI-generated masks using filtering techniques.

As a general-purpose segmentation model, SAM seems reasonably good at picking out objects when queried with points. However, this model might not be effective on complex images. For instance, food items often consist of different ingredients scattered across the dish.

Thus, our research questions will be as follows:

1) **Can we adapt a general-purpose multimodal model for food image classification with minimal training?**
2) **Are the segmentation maps created by explainability techniques good enough to guide general-purpose segmentation models?**
3) **Can we improve on the maps generated by XAI?**
4) **Are general purpose segmentation models good enough for food image segmentation?**

1.4 - Scope and Limitations

The purpose of this thesis is to explore image segmentation for food. While we could add additional datasets to benchmark our method's generalization on general tasks, we decided to stick with food datasets only. Our two datasets focused on dish-level and ingredient-level image segmentation, respectively. We used explainability methods to generate suggestions for what to segment. We do not explore the use of explainability methods beyond this.

We chose to use CLIP with CNN models to generate suggestions for what to segment. The reasons for this were that we found a library called OpenCLIP that provided a streamlined interface we could use, and that the explainability methods we ended up using lent themselves well to CNNs (Ilharco et al., 2021). We looked into various ways to use Grad-CAM and other explainability methods with vision transformers, but we had issues integrating them and decided to pivot. Thus, our choice ended up focusing on CNN-based models. However, it should be possible to integrate other multimodal model architectures.

The scope of this thesis is also to explore methods that require minimal training. This turned out to be a limitation, because the performance of our label selection component did not perform well. We had read that CLIP embeddings could be used for k-NN (k-Nearest Neighbors) search, but this approach did not work well for our datasets. While we tried adapting to the problem through lightweight training, the embeddings created by CLIP had many dimensions and we likely ended up overfitting to our datasets.

1.5 - Research Methods

This thesis will primarily follow the paradigm of quantitative research methods. We hypothesize about how a particular system might be created and what components go into it. To validate or reject our hypothesis, we employ statistical hypothesis testing methods.

1.6 - Main Contributions

The main contributions in this research work consist of:

- A proposed framework on how to add new modalities to existing segmentation models.
- A thorough study to measure possible configuration choices in our system.
- Suggestions for future work to improve our framework.

1.7 - Thesis Structure

The rest of this thesis is organized as follows.

Chapter 2 provides relevant background information for understanding the content of the thesis, including general information on image segmentation, explainable AI, dimensionality reduction, and our choice of experimental design and analysis.

Chapter 3 provides a literature search aiming to find sources that provide relevant foundation and context for this thesis.

In chapter 4, we present our method and the components that make it up. We also talk about how we study the impact of the hyperparameters that affect the components.

In chapter 5, we show the results from our research. We present quantitative evaluations for our label selection component, as well as our results on the various datasets we test against. We show our test results and compare them against SotA. Finally, we present the results from our hyperparameter impact studies.

In chapter 6, we discuss our findings.

Finally, in chapter 7, we conclude our work with a summary and suggestions for future work.

2 - Background

This background chapter provides context for the rest of the thesis. We present a few concepts that might be new to the reader and add some context for image segmentation, explainable AI, our choice of experimental design and analysis. Dimensionality reduction is also mentioned, with an explanation of how we might use it.

2.1 - A Historical Perspective of Deep Learning

Since its inception in the mid 1900s, AI (Artificial Intelligence) has gone through multiple periods of hype and despair called AI summers and AI winters (Thorwirth, 2021). The methods applied have been many and the camps within artificial intelligence advocate for their preferred methodologies (Smolensky, 1987). One such approach was neural networks.

In the early days, neural networks showed promise for the way it set out to imitate the highest form of intelligence we know of: the human brain. Neural networks are typically called a connectionist method for the way they *connect* simple mathematical operations to be more than the sum of their parts. While some progress was made, neural networks were often disregarded for practical applications because they didn't scale well. The amounts of compute necessary to perform even simple tasks kept them from being widely applied.

However, in 1965, Chekhov's gun was loaded when Gordon E. Moore coined the eponymous Moore's Law, which states that the density of transistors in microchips grows exponentially, doubling at regular intervals (Schaller, 1997). It took until 2012 for the foreshadowing to pay off, when Krizhevsky et al. showed the world how neural networks finally had the necessary compute to outperform contemporary methods on image classification (Krizhevsky et al., 2012). By applying two high-end consumer GPUs (Graphics Processing Unit), his team trained their AlexNet model on ImageNet. AlexNet outperformed the next best competitor by 10.8 percentage points, a relative improvement of 41%, when competing against contemporary models on the ImageNet competition ILSVRC-2012.

Since 2012, both industry and academia have recognized the potential of neural networks, and today they are one of the preferred methods for achieving human-level tasks. Neural networks are growing ever more complex, with some of the largest models having trillions of parameters, and more than 100 layers (Naveed et al., 2024). The subset of machine learning dealing with neural networks with multiple layers is called deep learning.

In just a few years, artificial intelligence went from being science fiction to reality. What previously seemed impossible is now commonplace. Nowadays, artificial intelligence even surpasses the average human on several tasks (Kiela et al., 2021).

However, training these models can be time consuming and expensive. Even when using pre-trained models and transfer learning, there are still costs involved. For instance, you might need data to adapt the model to a new task, domain or modality. There is also the cost of human labor involved. We hope our method can serve as a simple way to circumvent

this problem when one wants to add new modalities to their model.

2.2 - Image Segmentation

Computer vision tasks come in a hierarchical system of difficulties. The easiest is to look at an image and see if there is something in it. For instance, if the image contains a cat? The next step beyond this is, can you draw a box around the cat? These tasks are relatively simple and do not involve a large number of output variables. However, if you look at the whole image and want to know each individual pixel that contributes to a cat, that is more difficult. This task is called image segmentation.

2.2.1 - Types of Image Segmentation

There are multiple types of image segmentation, respectively called semantic segmentation, instance segmentation and panoptic segmentation (Hao et al., 2020). Figure 1 shows an example of various types of segmentation.

Semantic segmentation is when you look at an image and want to label every object in the image, based on the type of class it belongs to. For instance, you might have an image containing two cats. Semantic segmentation would give a common mask to the whole image, saying that both cats are part of the same semantic class.

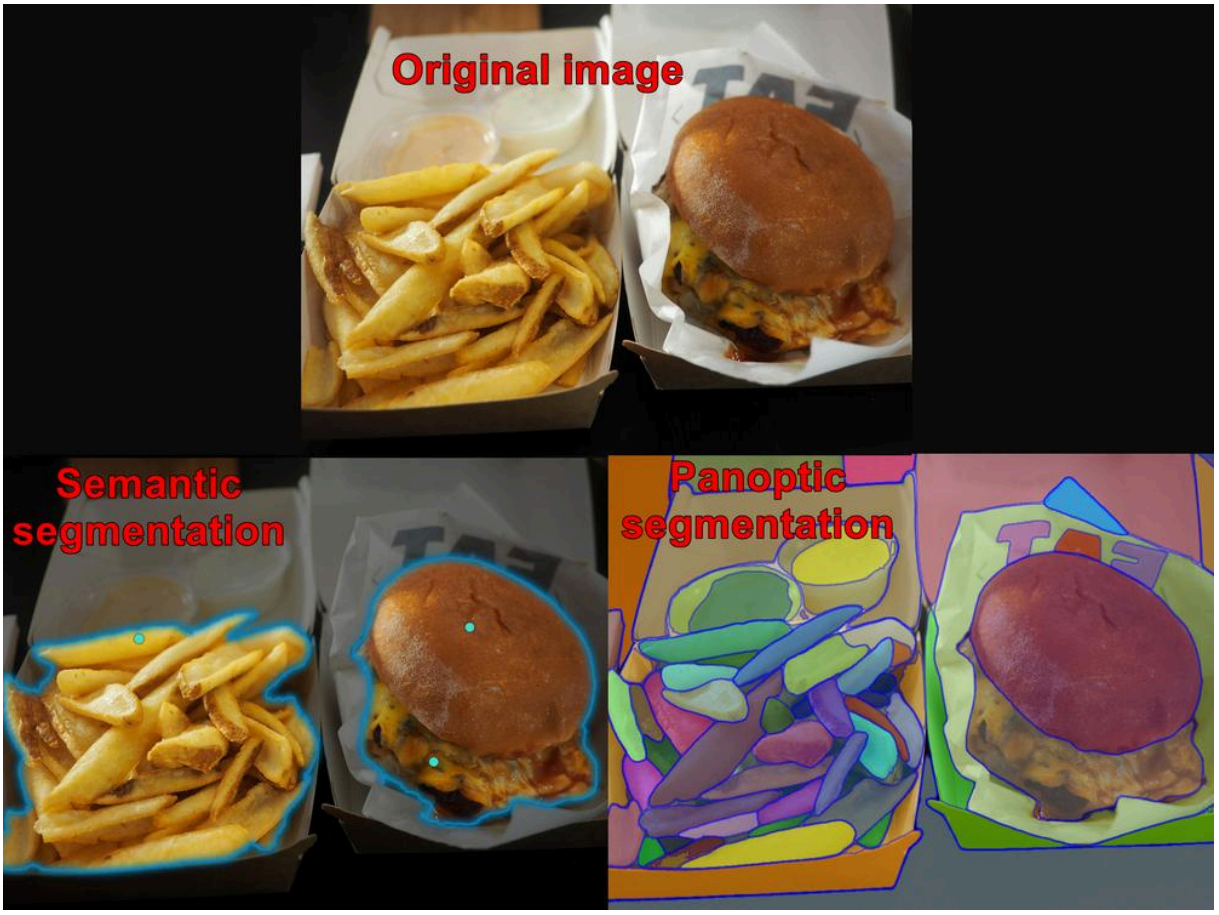
Next up, we have instance segmentation. Instance segmentation is interested in seeing an image as a collection of objects. Each instance of an object in this image will receive its own segmentation mask. So if you have an image with two cats, they will be separate entities with separate masks.

Lastly, we come to panoptic segmentation. Panoptic segmentation, meaning all-seeing, is different. This segmentation type will look at the whole image at the same time and will try to give segmentation masks to every pixel in the same image, according to which object it belongs. It does this by assigning both a class and an object id to every pixel.

In this thesis we will primarily focus on semantic segmentation.

Figure 1

A visualization of different segmentation types



Note. Top: An image of a hamburger and fries. Bottom left: Semantic segmentation of food in the image. Bottom right: Panoptic segmentation of all elements in the image. Own work adapted from a Creative Commons licensed image. From: Wikimedia Commons [Image], by JIP, 2016, Wikimedia Commons. https://upload.wikimedia.org/wikipedia/commons/d/d1/Fat_%22L%22_Half_Pound_Burger_at_home.jpg CC-SA 4.0

2.2.2 - Image Segmentation Evaluation

We decided to use mIoU (Mean Intersection Over Union) and aAcc (Pixel accuracy) as evaluation metrics for this thesis. This is because most of the papers we found that work with FoodSeg103 have used those metrics, making them easy to compare against (Blumenstiel et al., 2023; Lan et al., 2024; Wu et al., 2021). For other datasets, researchers might have converged on other metrics

Papers have differing opinions on what metrics to use. For instance, in (Busra Emek Soylu et al., 2023) argues that aAcc and mAcc are rarely used for segmentation, without further comment. In contrast, (M. Zhang et al., 2020) call them commonly used metrics.

The important thing to note is that one should use metrics with a purpose. The first thing to consider is what existing papers have converged on, because we use them as a way

to compare results with other researchers. We should also consider what metrics make sense for the task and dataset we are working with, both to analyze our results and for communicative purposes.

2.2.2.1 - A Note on Boolean Operations

When working with semantic segmentation evaluation, you typically have two masks: the suggested and the ground truth. These are like images where the value is either black or white and nothing in between. You can then overlay this mask onto your original image to see what is segmented out for a specific class. If you ask the model to segment out anything that is a cat, everything in the image that is recognised as cat will be white in the segmentation mask, and everything else, like other objects and the background, will be black. For each class you are interested in finding, you will have one of these masks.

As mentioned, these masks have either values of zero or one at each pixel.. You have two distinct and different states, meaning each pixel in these matrices is a Boolean variable. That is a property we will make use of. You can also think of each of these cells in a matrix like a set whose values are either zero or one. That means we can perform set operations like intersection and union. Later in this subchapter, we will use the overlap between the ground truth segmentation mask and the model's predicted segmentation mask.

2.2.2.2 - aAcc

aAcc is also called pixel accuracy (Wu et al., 2021; M. Zhang et al., 2020). When comparing two segmentation masks, it looks at the correspondence between the two values at any pixel location. At that pixel location, the accuracy is 100% if they are the same. If the pixels are not the same, the accuracy is calculated as 0%. We then average this over the whole image to generate the pixel accuracy.

Accuracy can often be a misleading metric. It has its benefits, but the problem is that you can have a high accuracy and the model can still perform badly. An analogy for this is, if you were to predict who has cancer: if you say nobody has cancer, you will probably be correct more than 99% of the time. However, this is not a useful way to screen patients, since the patients who have undiagnosed cancer will not discover if they have cancer. Because of this, we should supplement accuracy with other metrics.

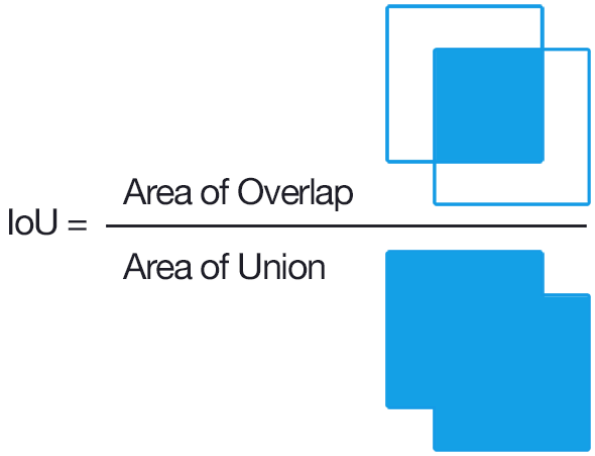
2.2.2.3 - mIoU

mIoU (Mean Intersection Over Union), also called the Jaccard Index, is the average IoU over all segmentation classes (Wu et al., 2021; M. Zhang et al., 2020). This is shown visually in Figure 2. The metric looks at the segmentation mask that the model generates and compares it to the ground truth segmentation mask.

The reason for using mean intersection over union in addition to accuracy is that it is better at showing if we actually found the objects we are looking for inside the image and generated a good segmentation mask.

Figure 2

Intersection over Union shown visually



Note. This figure visually represents the IoU metric. From: Wikimedia Commons [Image], by Adrian Rosebrock, 2016, Wikimedia Commons.
https://commons.wikimedia.org/wiki/File:Intersection_over_Union_-_visual_equation.png CC-SA 4.0

2.3 - Explainable AI

Artificial intelligence is becoming better and better for two main reasons. First, we generate more data than ever before. Everything and everyone is now connected to the Internet, leading to continuous data logging. In big data this is typically called the four Vs: velocity, veracity, volume, and variety (Addo-Tenkorang & Helo, 2016). People barely think twice about uploading their videos and pictures to YouTube and Facebook, and they are willingly describing their contents. The first reason is thus the increasing amount of data.

However, data is not useful without a way to analyze it. This leads to the second point: the more data we have, the more computational power is needed to analyze it. As mentioned previously, Moore's law has facilitated this by facilitating increased computational power. We are able to build ever more advanced models that perform better and better.

However, as our models become more advanced, they become increasingly difficult to interpret and understand. Moreover, how can we explain the inner workings of these complex models to someone? Under the GDPR (General Data Protection Regulation), if an algorithmic decision is made about a customer, an explanation for that decision is legally required if the customer requests one. Luckily, there are subfields within AI that deal with

understanding these models.

XAI (Explainable AI) and interpretability are big subfields that aim to make artificial intelligence more understandable (Barredo Arrieta et al., 2020). However, they have distinct focuses.

Interpretability refers to the extent to which a human can understand the cause of a decision made by a model. It implies simplicity and clarity in the model's functioning, allowing users to easily grasp how inputs are transformed into outputs. As mentioned in (Barredo Arrieta et al., 2020), "[Interpretability] is defined as the ability to explain or to provide the meaning in understandable terms to a human."

XAI, on the other hand, goes a step further by developing methods and techniques designed to provide detailed insights into complex models. XAI techniques strive to explain how and why a model makes certain decisions, even when the model itself is inherently complex, such as deep learning networks. This can involve creating surrogate models, visualizations, or feature importance scores to shed light on the inner workings of these sophisticated systems. (Barredo Arrieta et al., 2020) says that "explainability is associated with the notion of explanation as an interface between humans and a decision maker that is, at the same time, both an accurate proxy of the decision maker and comprehensible to humans".

We do not need insight into the inner workings of deep learning models, so interpretability is less of a requirement. What we do need is XAI, as it can be an interface for communication between the user through text prompts, and the models. In the literature review chapter, we will further explore specific explainability methods for deep learning.

2.5 - Design and Analysis of Experiments

In engineering, we are often concerned with the optimization of processes and symptoms to achieve optimal performance. This often involves the use of experiments to collect data, which we later analyze to reach conclusions. DoE (Design of Experiments) provides a structured way to accomplish this by enabling us to plan, conduct, analyze and interpret tests efficiently and effectively (Guthrie, 2020).

Given a set of input variables and output responses, we wish to discover how the inputs affect the outputs. DoE helps uncover the relationships between the inputs and outputs by offering suggestions on how to set up our experiments. We often test a single input variable at the same time, which is called an OFAT (One-factor-at-a-time) experiment, such as the effect of a drug against placebo. However, there are experiment designs under DoE that allow us to test multiple variables at the same time. Experiment designs that allow for multiple input variables can be more efficient and require fewer trials.

2.5.1 - Objectives

There are multiple criteria we can use to choose an appropriate experimental design. In (Guthrie, 2020), Table 3.1 shows their design selection guideline. Their guideline suggests choices based on the number of factors we wish to include and the type of design objective one is working with. They separate between comparative objectives, screening objectives, and response surface objectives.

In a comparative objective, the main question is whether a variable you think is important causes a significant change in the response at different levels. In a screening objective, you might have many effects, but you want to find out which of them are the most important ones. In an experiment with a response surface objective, we want to estimate the shape of a response surface.

In our case, we want to run a study to find out what hyperparameters are the most important for our model. This would be a screening objective, since we are most concerned with the strength of each individual hyperparameter.

2.5.2 - Experimental Designs

Once we know what objective we have, we can use the design design selection guideline to select an appropriate design. For a screening objective with multiple variables, (Guthrie, 2020) suggests using either a full or fractional design, or a Plackett-Burman design.

In a full factorial design, we list all the possible combinations of options we could test. If we have k factors and want to test $n$levels per factor, we would need to run n^k trials. Unfortunately, this grows very quickly.

Luckily, we often don't need to run all those experiments. With a fractional factorial design, we use one of various strategies to run a fraction of the full number of experiments called for by a full factorial design.

For our purposes, we landed on using Taguchi designs as our experimental design of choice (Guthrie, 2020; K. Krishnaiah & P. Shahabudeen, 2012; Peter J. Woolf, 2009). Taguchi designs rely on orthogonal arrays to reduce the number of required trials. Orthogonal arrays are a way to design experiments in such a way that the values of each variable occur evenly across all possible combinations, ensuring that all levels of each factor are tested independently of the others. This allows for efficient and balanced experimental designs, which are useful for studying the effects of multiple factors simultaneously. In the end, we landed on an L16B design with five variables and four levels per variable (*L16b Orthogonal Array*, n.d.). Running a full factorial design would have meant $mn^k = m \cdot 4^5 = 1024m$ experiments per trial, with m determining how many times to repeat the trial. With the L16B array, we can do the same experiment in $16m$ trials.

2.5.3 - Analyzing experiments

To analyze the experiments, we employ multiple techniques. When we keep a single variable constant in the data generated under the Taguchi experimental design, the variability introduced by the orthogonal arrays will likely generate non-normal data.

First off, we calculate confidence intervals using bootstrapping (Devore et al., 2021). This ensures we end up with robust confidence intervals for the population mean, despite limited observations and non-normal data.

Next, we can use permutation tests to test our hypotheses (Devore et al., 2021). Permutation tests are non-parametric and convenient when we don't know the shape of the distributions we want to compare. This makes them a good alternative to t-tests and ANOVA, which relies on the data being (approximately) normally distributed (Devore et al., 2021).

When analyzing the difference between two variables, he will use the null hypothesis H_0 that the variables have the same impact on the response variable, with the alternative hypothesis H_a being that the variables affect the response variables differently. We set a threshold at $p = 0.05$ as our cutoff for statistical significance.

2.5 - Dimensionality Reduction

DImensionality reduction can be a good first step in data preprocessing. When dealing with high-dimensional data, dimensionality reduction aims to reduce the number of dimensions while keeping as much of the useful information as possible. Three techniques that we use to varying levels during this thesis include PCA (Principal Component Analysis), t-SNE (-Distributed Stochastic Neighbor Embedding), and UMAP (Uniform Manifold Approximation and Projection) (Jolliffe & Cadima, 2016; Maaten & Hinton, 2008; McInnes et al., 2020).

2.5.1 - PCA

PCA (Principal Component Analysis) is a linear dimensionality reduction technique that transforms the data into a new coordinate system (Jolliffe & Cadima, 2016). The new coordinates are called principal components, which are orthogonal to each other and ordered by the amount of variance they capture from the data. The primary goal of PCA is to reduce the dimensionality of the data while preserving as much variability as possible.

A few advantages of PCA is that it is relatively quick to compute, especially compared to t-SNE and UMAP. Being a linear method, PCA will keep linear properties of the data it operates on. However, it is not good at capturing non-linear relationships, and thus might not fit every problem.

2.5.2 - t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear dimensionality reduction technique that is particularly well-suited for visualizing high-dimensional data. t-SNE reduces dimensions by converting similarities between data points into joint probabilities and then minimizing the divergence between these probabilities in the high-dimensional and low-dimensional space (Erdem, 2022).

2.5.3 - UMAP

Uniform Manifold Approximation and Projection (UMAP) is another non-linear dimensionality reduction technique that is faster and often more effective than t-SNE for certain datasets (*How UMAP Works — Umap 0.5 Documentation*, n.d.; *Understanding UMAP*, n.d.). UMAP is based on manifold theory and topological data analysis, aiming to preserve the global structure of the data while maintaining the local neighborhood relationships.

Some benefits of UMAP over t-SNE is that it's both faster when the number of dimensions grows and better at preserving the data's global structure (Altin & Cakir, 2024; Dalmia & Sia, 2021).

2.6 - Summary

In this chapter, we discussed essential topics to provide a foundation for understanding the thesis. We introduced deep learning and explained why it has become a preferred method for image analysis in the past decade. We also explored image segmentation, covering different types and evaluation methods for segmentation models. Additionally, we explored various aspects of XAI (Explainable AI), such as interpretability and explainability. We also touched on dimensionality reduction and experiment design.

With this background information in hand, we now move on to the literature review chapter, which examines relevant research for the proposed methods in this thesis.

3 - Literature Review

Having looked at the theoretical background necessary for the thesis, we continue by reviewing the existing literature. In this chapter, we intend to discuss literature to bring context and perspectives to our idea, and to understand the ongoing conversation within food image segmentation.

Relevant sections include searching for relevant datasets, multimodal embedding models, general image segmentation methods, food-specific image segmentation methods, XAI methods, as well as the interaction between XAI, multimodal embedding models and segmentation models.

3.1 - Food Image Segmentation Datasets

There are multiple research datasets available for food image tasks (Konstantakopoulos et al., 2024). They can differ in the task to be performed, such as classification, segmentation, recipe generation, geolocalization, volume and calorie estimation, and more. There is also variation in what to do, like the difference between dish-level and ingredient-level tasks, single-item and mixed-item annotation, and possibly hierarchical relationships such as recognizing that french fries is a potato dish. Furthermore, datasets might contain a specific type of food, like fruits, and they can differ in regional origin. Last, but not least, the origin and care put into the datasets differ. Some authors choose to create their own datasets by hand while others rely on labels from either other datasets, image search engines, or crowdsources annotations. This means that the quality of the datasets vary. For instance, some datasets seem large, but their segmentation masks might be automatically generated by another machine learning model of low quality, and haven't gone through proper quality control afterwards.

The two major tasks that are relevant for this thesis are segmentation and classification. Other types of datasets could also be relevant for this thesis. For instance, some datasets have enough images to consider self-supervised training methods. Recipe annotated datasets might also contain ingredient-level annotations, if extracting them to a relevant format is possible.

During our initial review, we read and taxonomized multiple review papers of available datasets (Konstantakopoulos et al., 2024; W. Wang et al., 2022; Wu et al., 2021). The authors provided tables with information about the various datasets, including publication year; dataset tasks; number of annotation classes, dishes and ingredients; the number of images in the dataset; and other information. From the taxonomy, we found nine applicable food image segmentation datasets.

However, several of these are not available for public download. The first problem is that not all authors publish the datasets publicly, which could be due to copyright and licensing constraints.

Our inclusion criteria is that the dataset should be a segmentation dataset of

sufficient size. We initially also considered looking at classification datasets for training and fine-tuning classifiers. Due to many of the datasets being imbalanced and having different data distributions, we decided to limit the scope to segmentation only. The review papers we found (Konstantakopoulos et al., 2024; W. Wang et al., 2022; Wu et al., 2021).

3.1.1 - Food50Seg

Food50Seg is a dataset for food segmentation, consisting of 5,000 images of fifty different dishes (Aslan et al., 2020). Food50Seg is based on an unnamed dataset the authors refer to as Food50Chen[1], after the first author of (M.-Y. Chen et al., 2012). Food50Chen itself is based on 5,000 base images of mostly Asian food. The dataset has 50 dish-level labels, with 100 images per label. Through image augmentations, (M.-Y. Chen et al., 2012) expanded the dataset to 120,000 images, but (Aslan et al., 2020) used only the 5,000 base images for Food50Seg.

(Aslan et al., 2020) had two people manually create segmentation masks for Food50Seg. They tested 10 different models on the dataset to segment both the presence and absence of food. The best model, GUN, achieved an mIoU of 0.891 and what we believe is the pixel accuracy of 0.944. For semantic segmentation of the different dish-level annotations, the same model achieved an mIoU of 0.758 and a pixel accuracy of 0.890. In addition to this, they tested how various types of noise would impact the segmentation model performance, such as Gaussian noise, Gaussian blurring and JPEG compression. Most of their tested models fared well against JPEG compression, but both Gaussian noise and blurring had a considerable impact, dropping their dish-level segmentation mIoU scores by 20-30 percentage points.

We were able to get in touch with the authors of Food50Seg, who gave us access to their segmentation masks (Aslan et al., 2024). We also got access to the underlying Food50Chen dataset (M.-Y. Chen et al., n.d.).

3.1.2 - Mixed Dishes

In their original paper, the authors presented two datasets at the same time. For simplicity and consistency, we use the same name as (Konstantakopoulos et al., 2024) and call it Mixed Dishes.

The authors behind Mixed Dishes say that “Mix dish recognition, whose goal is to identify each of the dish types presented on one plate [sic]” (Y. Wang et al., 2019). The authors present 10 different datasets they see as possible candidates for such a mixed dish dataset and taxonomize their contents. However, The other available datasets did not consistently have multiple foods and mixed dishes in the same image. In the end, it seemed

[1] Not to be confused with Food50, which is a separate dataset with a similar name.

the authors decided to make their own datasets: Economic Rice and Economic Beehoon.

The Economic Rice dataset has 9,254 images and 164 dishes. It was collected using cell phone cameras and labeled by seven students. The authors mention the images can be quite challenging. The images have overlapping dishes with no clear boundaries, And even the same canteen can make the same dish look different.

The Economic Beehoon dataset contains 2,851 images of beehoon, which is a popular Singaporean food type that's typically popular for breakfast. From what we gather, beehoon[2] Is often translated as a rice vermicelli and is a thin type of noodle. In the context of a dish, beehoon seems to combine rice with multiple ingredients. The dataset has 54 different classes which the authors say are dish categories.

The authors tried training models to recognize the dishes in the two datasets separately, using Inception-based models. They tried training models either with pre training or from scratch. Additionally, they also tested negative sampling. They report results using precision, recall and F1 scores.On both datasets, they reached the highest results with a pre-trained Inception-V4 model using negative sampling. On the Economic Rise dataset, the highest F1 score was 0.498, and they reached an F1 score of 0.571 on the Economic Beehoon dataset.

We were not able to find the dataset available for public download.

3.1.3 - SUEC Food

The SUEC Food dataset consists of 256 different segmentation classes and a total of 31,395 segmented images (Gao et al., 2019). Originally, UEC Food-256 had bounding boxes. (Gao et al., 2019) added segmentation masks to create SUEC Food dataset. They do this by manually segmenting 600 images and use GrabCut to generate the remaining segmentation masks (Rother et al., 2004).

In the paper, the authors test their own models against the datasets. Their models are built on FCNs (Fully Convolutional Networks) for segmentation (Jonathan Long et al., 2015). All their MFCN approaches slightly outperform the baseline FCN, which has an mIoU of 0.9143, as well as the underperforming Grabcut@5 with an mIoU of 0.7730. Their best result was with their MFCN-B model, with an mIoU of 0.9210. They additionally attempt food volume estimation on the dataset.

The authors made their dataset available for download from Google Drive, with the link available at (Gao et al., 2019).

[2] According to the number of Bing Search results returned, the favored spelling might be bee hoon.

3.1.4 - Food201-Segmented

Food201-Segmented is a 2015 dataset from the paper Im2Calories, which was co-authored by Google (Meyers et al., 2015). It has 201 separate dish classes of meals typically found in a restaurant, with 10,100 training images and 2,525 testing images. Google used crowdsourcing to extend the Food101 dataset by another hundred categories, creating Food201-Multilabel. This new dataset had roughly 50,000 images, from which they used 12,625 to create a new dataset they called Food201-Segmented.

To perform calorie estimation, Google also used depth maps and trained convolutional neural networks for this task. This way they could estimate the volume the food actually took up inside the image.

The authors released Food201-multilabel, the classification dataset that Food201-segmented is based on. However, we were not able to find Food201-segmented publicly available for download.

3.1.5 - UECFoodPix

UECFoodPix is a dataset from 2019, which extends the UEC-Food100 dataset by automating dish detection and segmentation mask generation using a combination of methods (Ege et al., 2019; Matsuda et al., 2012). Using 102 types of dishes, the authors generated dish-level annotation masks for 10,000 images, and provided 9,000 images as training data and 1,000 images as testing data. Since UEC-Food100's bounding boxes covered the whole as one dish instead of instance-level food items, they had to manually recreate bounding boxes for all the images. To generate the segmentation masks, they then used GrabCut on the images (Rother et al., 2004).

In the same paper, the authors proposed a model for calorie estimation which used YOLOv2 trained to detect rice in images, and then used a VGG16-based U-Net model to segment out rice in the image (Redmon & Farhadi, 2017; Ronneberger et al., 2015; Simonyan & Zisserman, 2015). They used this segmentation mask to estimate the amount of rice present in the picture and converted that estimate to a calorie estimate.

UECFoodPix is publicly available and can be downloaded without issues (*UECFoodPix,UECFoodPixComplete*, n.d.).

3.1.6 - UECFoodPix Complete

UECFoodPix Complete is a 2021 paper by new authors that attempts to improve on the original UECFoodPix (Ege et al., 2019; Okamoto & Yanai, 2021). The problem with the original method is that they generated segmentation masks automatically using GrabCut. (Okamoto & Yanai, 2021) went over the 10,000 images and tried to improve the

segmentation masks through manual review and correction.

Results improved compared to the original dataset. They saw the performance improve even with partial hand annotations. However, when they had fixed the whole dataset they saw dramatic improvements. The authors trained what was at the time a state of the art semantic segmentation model called DeepLab V3+. When they compared the model trained with the two datasets, the accuracy went up from 56.0% to 66.8% and the mIoU had improved from 41.6% to 55.5%. These results imply that accurate segmentation masks are important to train high-quality segmentation models.

UECFoodPix Complete is publicly available and can be downloaded without issues (*UECFoodPix,UECFoodPixComplete*, n.d.).

3.1.7 - FoodSeg103

FoodSeg103 is a dataset from a paper published in 2021 (Wu et al., 2021). It aims to provide images and segmentation masks for both dishes and the ingredients those dishes consist of. They mention two drawbacks of previous datasets. First, they say that there's "[...] a lack of high quality food image datasets with fine-grained ingredient labels [...]" and that "[...] existing datasets either carry coarse ingredient labels or are small in size". The second point is that "[...] the complex appearance of food makes it difficult to localize and recognize ingredients in food images [...]". Thus, the authors built two new datasets called FoodSeg103 and FoodSeg154. FoodSeg154 contains 9,490 images of food with 154 ingredient classes. FoodSeg103 is a subset of FoodSeg154 because of licensing issues and contains only 7,118 images and 103 ingredient classes.

The authors provide a list of 16 different datasets as previous works and comment on some of them. Most of the listed datasets contain either classification or recipe annotations, and can not directly be used for segmentation. However, they list and comment on UECFoodPix and UECFoodPixComp. They argue that the main benefits of FoodSeg103 is that it contains a considerably larger number of segmentation masks, that they have ingredient-level rather than dish-level annotations, and that their fine-grained annotations can be used to analyze food nutrition and calorie estimation.

They also note that it's a more challenging dataset when evaluating the UECFoodPix variants against FoodSeg103. So one reason for this is that when ingredients are cooked, they might look similar to each other. Yeah. Cooked pineapples and cooked potatoes can look very similar depending on the dish. If you're looking at a dish, you might be able to recognize something as a particular dish because of, or in spite of the ingredients that make it up. But when trying to recognize individual ingredients, this might not be the case. For instance, the model might learn what dish it's looking at, and try to discern what ingredients would typically be part of that dish, instead of looking at a particular ingredient in isolation.

FoodSeg103 is publicly available for download (Wu et al., n.d.).

3.1.8 - Discussion

Overall, datasets for food image tasks come in many forms. For food image segmentation specifically, several problems are present.

The obvious problem for several of these datasets is whether they are available for public use. We failed to find download sources and contact information for several of the datasets. This might be due to licensing issues, as mentioned in FoodSeg154 (Wu et al., 2021).

Some of these problems stem from the task itself. For instance, manually creating high-quality datasets can be expensive and time-consuming. Some tasks even require domain expertise and additional care. For a task like ingredient segmentation, the annotator needs to decide whether an ingredient might be a piece of cooked pineapple or potato (Wu et al., 2021). Even for dish-level annotations, there is likely to be some overlap. Two stews might look visually similar due to the ingredients and preparation methods used, and one would have to taste or smell the dish to be sure.

Other problems come from the data preparation methods used. Some of the available datasets were created with automated segmentation mask generation methods. Thus, training and evaluating on the dataset might pick up on the failures and biases of the original data generation method.

In the end, we chose FoodSeg103 and UECFoodPix Complete as our preferred datasets. They meet the requirements of being publicly available, manually annotated, and easy to work with. We also got in touch with the authors of Food50Seg, but had problems integrating it due to time issues.

3.2 - Multimodal Image Models

Multimodal image models combine various modalities, like text and images (Baltrušaitis et al., 2017; Parcalabescu et al., 2021). For instance, models like Stable Diffusion and DALL-E lets you generate previously unseen images by just writing a text prompt to the model (Ramesh et al., 2021; Rombach et al., 2022).

The general idea for this thesis is to use visual models in combination with explainability techniques. We want to find the regions in our image that contain the class we want to segment. Depending on the amount of data and resources we have available, it might not be possible to train a visual model for this task. Thus we want to see if it's possible to use multimodal models as an interface for prompting.

If we have a shared embedding space for the image and another modality, we can use explainability techniques to find regions of interest. In this chapter, we will explore a couple

of these models and decide which one lends itself best to the task.

3.2.1 - CLIP

CLIP (Contrastive Language-Image Pretraining) is an approach that trains separate models to align semantically rich outputs in embedding space (Radford et al., 2021). This means that given a semantically similar input, like an image and a text description of its contents, the outputs should be fairly similar. In mathematical terms, we can calculate the distance between them using an appropriate distance metric, like the cosine distance or euclidean distance.

An application of this might be as an image search engine. Previously, such a search engine might look at the text descriptions letter by letter to find a match. It might read webpages and use the surrounding text to guess what is in the image. Using CLIP, you can instead search for specific objects inside an image. For instance, you could use this to search through a movie to create time stamped content warnings for those with a phobia of spiders.

In (Radford et al., 2021), the authors start by generating text and image embeddings using a pre-trained model for that modality. Then, to align them, they project the embeddings to another embedding space by multiplying with a weight matrix. They then take the dot product of the two embeddings to create the predicted logits. The logits are then compared to the actual labels using the cross entropy loss function, and the error gradients are backpropagated so the network can learn. To summarize, they compare how different the embeddings from each modality is and try to minimize the difference.

3.2.2 - ImageBind

ImageBind is an idea from a paper that intends to combine *multiple* modalities in the same embedding space (Girdhar et al., 2023). It operates on images, videos, text, audio and more. Their method pulls data from different modalities not represented by their training data, into the same embedding space. This allows the models to perform zero-shot and cross-modal retrieval between unrelated modalities even when the training does not directly represent this relationship.

Some modalities are not inherently connected by the methods they use. However, the authors show that shared embedding spaces give rise to emergent alignment between modalities. For instance, they don’t have an explicit dataset with depth information for audio. However, by leveraging depth information from images, they can connect to video, which contains audio. Thus, by leveraging a shared embedding space, they can connect depth information to audio.

As far as implementation details go, the authors mention using contrastive loss to align the different modalities. They train on pairs of modalities using InfoNCE. Data of

different modalities are encoded using various models with modality-specific linear projectors to move it into the shared embedding space.

3.2.4 - LiT

Locked-image text Tuning is a method for training multimodal models for text and vision (Zhai et al., 2022). The authors claim that this approach outcompetes CLIP by a considerable margin on their selected benchmark, with nearly double the percentage points on certain zero-shot tasks for the ImageNet dataset. It seems that training results depend heavily on the training data they employ. Using a non-public dataset, they closed the gap on multiple tasks, compared to when they used a subset of the yfcc100m dataset for training.

The method uses separate models for text and vision. They propose two training schemes they call “Contrastive pre-training” and “Contrastive-tuning” respectively. Thus, the paper uses a contrastive loss function akin to what they use in (Radford et al., 2021), but they are unclear of the specifics beyond this.

Contrastive pre-training seems like a method to use pairs of vision and text to train models from scratch. They propose a method that uses two separate *towers* to create embeddings of the same size. However, we fail to see how this differs from the CLIP method introduced in (Radford et al., 2021), which is also referenced in the subsection on contrastive pre-training. It’s possible they employ

In their subsection on contrastive-tuning, they introduce the idea of locking, which is another term for freezing the weights of a particular model. Thus, each of the text and visual models can be trained in one of three methods: (L) locked pre-trained, which uses a pre-trained model with frozen weights, (U) unlocked pre-trained, where the pre-trained model can be changed during training, or (u) unlocked from-scratch, where the model weights are trained from a randomly initialized state. LiT refers to the Lu configuration, where the vision and text models respectively use the L and u configurations.

3.2.3 - Sigmoid Loss for Language Image Pre-Training

(Zhai et al., 2023) explores the use of what they call the pairwise sigmoid loss. They propose it as an alternative to softmax-based contrastive learning. According to their results, using the sigmoid loss allows for larger batch sizes overall and higher performance at smaller batch sizes. They also test and show that larger batch sizes don’t necessarily perform better. They pushed the batch sizes to one million, but found that smaller batch sizes of 32k were usually sufficient as the benefits of larger batch sizes quickly diminished.

Methods using the softmax-based contrastive learning can easily be adapted to use the sigmoid loss function instead. The authors adapted both CLIP and LiT, calling them SigLIP and SigLiT respectively (Radford et al., 2021; Zhai et al., 2022).

3.2.4 - Discussion

When it comes to multimodal models that utilize a shared embedding space for the modalities, variants of contrastive training seem like a popular choice. As long as we can use one modality as a label for the visual model, we can utilize explainability techniques to locate the area of interest in the visual model.

It seems that these frameworks typically aim to train at least a full model. However, contrastive training seems to pull data into the same embedding space. One could, for instance, freeze the text and visual models, and use contrastive methods to train a projector head on top of each. Another thing to note is that LiT seems like a case of cross-modal knowledge distillation (Tian et al., 2022).

As the progress of science marches on, the models seem to get better. LiT seems to do better than CLIP, for instance, and SigLIP/SigLiT might do better than LiT. However, there are other things to consider than just performance. Ease of use can improve development speed, for example. While HuggingFace is a great platform that hosts multiple models and weights, OpenCLIP stood out to us as a useful code library (Cherti et al., 2022; Ilharco et al., 2021). The community around it has collected various versions of CLIP trained on multiple datasets and have packaged all of them using a standardized interface.

3.3 - Segmentation Methods

3.3.1 - SAM

Segment Anything is a paper from Facebook AI Research (Kirillov et al., 2023). Their SAM (Segment Anything Model) is a foundation model which aims to perform promptable image segmentation. The idea is to use a prompt encoder and an image encoder to create inputs for a mask decoder. Based on the inputs, the mask decoder will try to create an appropriate mask for any input image. Some of the prompts Meta trained the model for include point queries, bounding box queries, and segmentation mask queries.

The authors collected a large, diverse dataset of one billion images that they call SA-1B. This dataset consists of 11 million images and more than one billion segmentation masks. The images were collected from licensed images they intended to release under an open source Apache 2 license. Because SAM is trained on such a huge and diverse dataset, it’s able to generalize well to unseen segmentation tasks.

3.3.2 - DINO

DINO (From “self-**di**stillation with **no** labels”) is a self-supervised method developed by Facebook AI Research, which aims to leverage large, unlabeled datasets through self-supervised learning (Caron et al., 2021). DINOv2 improves on DINO, with better performance in benchmarks and demos (Oquab et al., 2024)

The title of the (Caron et al., 2021)is “Emerging Properties in Self-Supervised Vision Transformers”, and the emergent property mentioned is that DINO ends up performing well as a semantic segmentation network. This segmentation property can be seen when treating the later attention layers in the network as saliency layers. They claim their self-supervised DINO ViT models produce clearer segmentation masks than supervised ViT and CNNs do.

The idea behind DINO is to use two networks - one teacher and one student. The network branches in two, with the student and teacher on separate branches. At training time, two copies of the inputs are created and separately augmented. For instance, for an image, this might be cropping, resizing and changes to color. The inputs are sent through the networks separately and an embedding is created for each. The teacher branch has a centering operation applied to it afterwards, where the mean is computed from the current batch and subtracted. At the end, both branches go through a softmax activation to create the final outputs of the network. When applying the softmax operation for the teacher, it’s additionally adjusted with a temperature to dampen the certainty of the outputs. This process of having a network learn by dampening the certainty of a teacher is typically called knowledge distillation, or in DINO’s case, self-distillation (Gou et al., 2021). After calculating the softmax log probabilities, the loss is calculated by comparing the outputs. The error is backpropagated through the student network only: the teacher teaches the student. Finally, at some regular interval, the teacher network copies over some of the student network’s weights using exponentially moving averaging. Once training converges, the student network is extracted and used as the final product of the model.

3.3.3 - Grounding DINO

Grounding DINO is an approach to make a promptable segmentation model by combining pre-trained DINO models with text queries, making it a zero-shot general-purpose image segmenter (S. Liu et al., 2023). Their idea extends (L. H. Li et al., 2022), whose GLIP model learned to create bounding boxes for images based on image-text-pairs.

3.3.4 - LLaVA-Interactive

LLaVA-Interactive is a method that allows for multimodal interaction by using an LLM as the interface (W.-G. Chen et al., 2023). It uses the LLaVA model that combines language and visual understanding. SEEM is used for image segmentation and it incorporates GLIGEN

to generate images.

3.3.5 - DeepLab

DeepLab is a family of models, with the latest version being DeepLabv3+ (L.-C. Chen, Papandreou, et al., 2018; L.-C. Chen, Zhu, et al., 2018). It has been used extensively for food image segmentation, including in (Battini Sönmez et al., 2023; Meyers et al., 2015). The method uses atrous convolutions, which are convolutions with upsampled filters, to control the resolution and enlarge the field of view. They use spatial pyramid pooling to segment objects at multiple scales. To reduce the effect downsampling has on localization accuracy, they employ CRF (Conditional Random Field) as a strategy.

3.3.6 - Discussion

Overall, there seems to be several good options for general-purpose segmentation. SAM seems like a method with decent performance and it has a nice interface to work with.

3.4 - Food Segmentation Methods

We previously covered general purpose segmentation methods. Of course, research has been put into creating food specific deep learning segmentation models. In this subchapter, we will go through and review a couple of them.

3.4.1 - ReLeM

ReLeM (Recipe Learning Module) is a method from (Wu et al., 2021), the same paper where the FoodSeg103 dataset was published. The method contains a vision encoder and a text encoder, which are trained together. Once the vision encoder has been trained, its outputs are decoded by a vision decoder. The decoder is what makes the segmentation masks for the specific image.

In their approach, the authors use different types of encoders and decoders. For the text encoder, they tested LSTMs and transformers. For the vision encoder, they tried two types: ResNet-50 and ViT-16/B. For the decoder performing segmentation, they used a variety of methods, including dilation based networks, feature pyramid networks and transformer based decoder methods. These were respectively CCNet, FPN and SeTR.

In their results section, they compare their methods to segmentation with the respective decoder only. For each of their tested methods, their ReLeM version beat the baseline by a few percentage points on both mIoU and mAcc. In all cases, the LSTM-based

encoder performed better than the transformer version. Their highest result was an MIOU of 43.9%, with a model size of 723M total parameters.

Finally, they show the performance of their model when adapted to a different domain. As previously mentioned, FoodSeg103 is a subset of the full FoodSeg154, which contains an additional 2,372 images of Asian food. The ReLeM module improves the performance of the baseline network both before and after fine-tuning to Asian food.

3.4.2 - FoodMask

FoodMask is a neural network model for food image segmentation (Nguyen et al., 2024). The method is both able to segment and count occurrences of food items within an image. On its provided benchmarks, the authors show that it balances speed and segmentation quality well. The method builds on feature pyramid networks that branches to separately work on food recognition, segmentation, and counting.

3.4.3 - FoodSAM

FoodSAM is a paper that explores the use of Segment Anything Model for food image segmentation (Lan et al., 2024). They comment that SAM doesn't give class-specific information to the generated masks and apply various techniques to label them. Their proposed solution has three components, namely SAM itself, a semantic segmenter and an object detector. They use the semantic segmenter network to create coarse semantic masks, then match those with the high quality masks generated by SAM.

3.4.4 - FoodLMM

FoodLMM is a large multimodal model that can perform various tasks on food images, such as segmentation and recipe generation (Yin et al., 2023). They train their LMM using LoRA to train specifically for food tasks. The LMM was then trained using visual instruction tuning, where text queries and food images are paired. They had to generate their own datasets, including FoodDialogues and FoodReasonSeg, by using GPT-4 to generate conversational data for them. In the end, their model is trained with various datasets with the goal of performing food segmentation, nutrition estimation, food visual question answering and more.

3.4.5 - Im2Calories

In addition to creating Food201-Segmented, (Meyers et al., 2015) also trained a

model on the dataset. They based this model on DeepLab, which was pre-trained on ImageNet and fine-tuned on Food201-Segmented. After training the model for food image segmentation, they went on to perform food volume estimation with it.

3.4.8 - Benchmarking algorithms for food localization and semantic segmentation

In addition to creating Food50Seg, (Aslan et al., 2020) also benchmarked various CNN architectures for food image segmentation. Out of their tested models, SSNet and GUN performed the best, with over 90% mIoU on Food50Seg.

3.4.9 - CANet

CANet (Cross Attention Network) is a 2023 paper that boasts impressive performance on UECFoodPix Complete, with an mIoU of 68.9% (Dong et al., 2023). They build their method on ResNet by removing the last downsampling layers. The proposed method combines ideas from CNNs with attention from transformers by sending information through two separate paths that they later combine. They call the different paths the cross spatial attention and the channel attention. They additionally use wavelet transformations to obtain low and high frequency details.

3.4.10 - GourmetNet

In GourmetNet, the authors introduce what they call a WASP (Waterfall Atrous Spatial Pooling) module (Sharma et al., 2021). Their network uses a ResNet backbone, and they refine the features using spatial and channel attention modules before sending the results to the WASP. When they released their work in 2021, they achieved state of the art performance on both of the segmentation datasets they tested their method on.

3.4.11 - Discussion

It seems like there's considerable competition within the food image segmentation community. Several papers have been released and new ideas are introduced constantly. CNNs were used a lot previously, but newer methods seem to learn toward transformers, or at least borrow ideas like attention. One of the methods we found, FoodSAM, has already tried one way of using SAM for food image segmentation. However, we did not find any papers that use SAM and XAI specifically for food image segmentation tasks.

3.5 - XAI Methods

3.5.1 - CAM

CAM (Class Activation Mapping) is a 2015 technique originally developed for providing explainability for CNN models (Zhou et al., 2015). The authors developed a method to highlight what parts of an input image contributed to a particular classification decision. Put in simple terms, if the model says an image contains a cat (the class activation), what visual features contributed to this decision and where are they located (the map)?

In CAM, the authors weigh the channels in the last convolutional layer's channels by how much they contributed to a particular classification decision. To use attributes from the convolutional layers further on in the network, the authors use GAP (Global Average Pooling) to reduce the 2D CNN layer down to a single value per channel. These are the values they use as weights for each channel. To classify what the image contains, they apply the softmax activation function to generate probabilities for each output class. To create the final class activation map, all the convolutional channels in the last layer are compressed into one and the values are averaged. Class activation maps can be thought of as a monochromatic image. In a class activation map, brighter regions highlight where the particular class occurs. When overlaid onto an input image, they can for instance show what parts of the image contributed to classify the image as a cat.

A drawback of the CAM method is the limitation in what networks it can be used on. As mentioned above, CAM applies GAP to the last convolutional layer and feeds this directly to a softmax activation function. However, several neural networks have architectures where this is not possible. For instance, they might have several fully connected layers beyond the convolutional layers, which makes it difficult to disentangle and attribute a CAM weight to a classification.

CAM has since gained a lot of traction, with multiple people building on the ideas introduced. Some of the variations include Grad-CAM (Selvaraju et al., 2020), Grad-CAM++ (Chattopadhyay et al., 2018), LayerCAM (Jiang et al., 2021), Group-CAM (Q. Zhang et al., 2021), ScoreCAM (H. Wang et al., 2020) and gScoreCAM (P. Chen et al., 2022).

3.5.2 - Grad-CAM

Grad-CAM (Gradient-weighted Class Activation Mapping) builds on the ideas from CAM by using the gradients calculated when using backpropagation. These gradients flow from the classification decision to all parts of the neural network. The main advantage of using gradients is that information about the classification result can be sent to any part of the neural network. Thus, Grad-CAM addresses the previously mentioned drawback of CAM, and can be applied to arbitrary network architectures.

3.5.3 - Grad-CAM++

Grad-CAM++ claims to improve on Grad-CAM by changing the formula used to calculate the class activation maps (Chattopadhyay et al., 2018). They back up their blame quantitatively by measuring for example how much the method drops in object localization confidence when parts of the image get removed.

3.5.4 - LayerCAM

LayerCAM is yet another method that works with class activation mapping (Jiang et al., 2021). Because of the low resolution of CAM in later CNN layers, class activation mapping methods suffer when it comes to pixel accurate object location. The authors try to generate more fine-grained object localization information by rethinking the relationship between the feature maps and their corresponding gradients. When they analyzed previous methods like GradCAM, they noted that the methods typically assigned as much weight to each location in the class activation map. They propose that shallow layers tend to capture fine-grain details, regardless of whether they are part of the background or the target object. They try to address this by assigning a separate weight for each spatial location in a feature map using the backward class-specific gradients. Finally, information from all the layers is combined linearly into a class activation map.

3.6 - Using CAM Methods On CLIP

Various methods have tried using CAM to explain the outputs of CLIP.

ReCLIP uses a combination of GradCAM and a region-scoring method, and seems to do better on the included benchmarks than pure GradCAM (Subramanian et al., 2022).

CLIP-ES improves weakly supervised semantic segmentation by introducing softmax into GradCAM to suppress non-target classes and backgrounds (Lin et al., 2023). They introduce what they call the CAA (Class-aware Attention-based Affinity) module, which is based on multi-head attention.

(Petryk et al., 2022) uses CLIP as a Vision-Language model to guide attention with language. They train a separate model for classification and compare the attention between this model and CLIP's text/image embeddings using what they call "attention loss".

(Y. Li et al., 2022) creates an image/text similarity. They linear projections and use two training objectives, one with contrastive loss and the other with mean average precision, to correct explainability in their setup. As part of the result, they generate an intermediate

similarity matrix, that is then used to generate segmentations.

3.7 - Using SAM With CLIP

(Kirillov et al., 2023) explored using zero-shot text embeddings from CLIP as input prompts, showing that the task is possible. They use them as inputs for the prompt encoder part of their approach. However, they don't perform a quantitative evaluation in the paper, so it's difficult to say how effective it might be. They also didn't release these models to the public, like they did with the other prompt variants.

ClipSAM is a method that uses both image and text input to CLIP, which are thought of as a rough result. The rough result is fed to SAM, which generates a suggestion for a segmentation. SAM's output is combined with the outputs from CLIP to create a refined result. They claim their method can do zero-shot anomaly detection (S. Li et al., 2024).

SAM-CLIP trains both CLIP and SAM together for multi-task distillation. This way, they're able to perform a variety of promptable tasks, like classification, and instance and semantic segmentation (H. Wang et al., 2023).

3.9 - Summary

In this chapter, we went through relevant research literature to understand food segmentation, CLIP and other multimodal models and XAI methods. We looked into how CAM has been combined with CLIP and with SAM, and we looked in ways for which SAM, CLIP and CAM methods had been applied in our proposed way. We didn't find anyone who had tried the same approach we propose.

Thus, we move to the methodology section, where we present our method.

4 - Methodology

In the literature review, we got familiar with existing research on food segmentation, multimodal models, XAI and image segmentation.

Now, in the methodology chapter, we will give an overview of our proposed solution. We introduce our system for XAI-guided food image segmentation, which consists of six modular components. Finally, we provide some information about how we intend to test the impact of individual hyperparameters on the overall system.

4.1 - System Overview

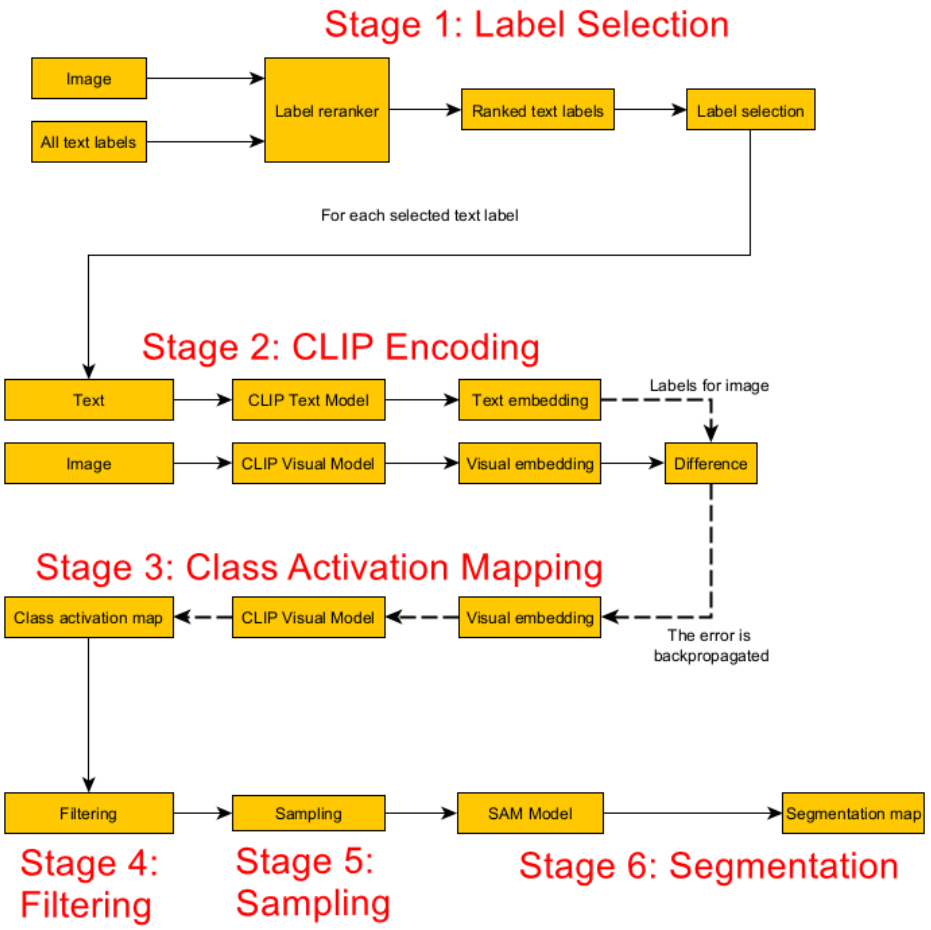
Our method is a system consisting of the six stages shown in Figure 3, each performing a vital task. The stages are: (1) label selection, (2) encoding, (3) class activation mapping, (4) class activation map filtering, (5) point sampling, and (6) segmentation mask generation. Each stage will be covered in its own subsection.

Each stage is a module that users can replace when better methods emerge. The only requirement for incorporating new modules is adherence to the module data passing contracts.

We chose to use CLIP as the multimodal backbone. Our code uses a library called OpenCLIP, which provides a streamlined interface to download and use multiple CLIP configurations and weights (Cherti et al., 2022; Ilharco et al., 2021).

Figure 3

A system diagram for our full method



Note. Our system consists of five stages in total. In stage one, we decide what object types to look for in the image. We compare and rank the images and text labels based on their similarity and then apply selection criteria to determine the object classes in the image. In stage two, we generate embeddings for the text and visuals. We treat the text embeddings as the ground truth label for the visual model. Then, in stage three, we can backpropagate the error through the visual model to generate a class activation map. In stage four, we post-process the class activation map. In stage five, we sample points from the class activation map. In stage six, we provide the sampled points to the segmentation model and instruct it to produce a final segmentation map.

4.2 - Label Selection

The first stage of our system is the label selection stage. In this stage, we attempt to classify the presence of food items to segment in the later stages of the system. Overall, we have tested multiple methods, including classifiers like k-NNs (k-Nearest Neighbors), SVCs (Support Vector Classifiers), logistic regression and XGBoost, as well as clustering-methods like agglomerative clustering, DBSCAN and k-means clustering (T. Chen & Guestrin, 2016; Ester et al., 1996; James et al., 2021). We also tested various neural net configurations.

We generated embeddings for both the image and the text representations of the target classes for the datasets. Some examples from FoodSeg103 include the phrases *rice*, *broccoli*, and *bread roll*. As mentioned in (Radford et al., 2021), the CLIP encoders are trained

using the dot product between the embeddings as the similarity metric. We tested using both the shared embedding space between text and images, as well as just the embedding space for images from the training datasets.

PCA (Principal Component Analysis) was also a method we looked into. This was an attempt to combat the high dimensionality of CLIP's shared embedding space, by training on only the embedding features with high variance.

We generated embeddings for both the image and the text representations of the target classes for the datasets. Some examples from FoodSeg103 include the phrases *rice*, *broccoli*, and *bread roll*. As mentioned in (Radford et al., 2021), the CLIP encoders are trained using the dot product between the embedding as the similarity metric.

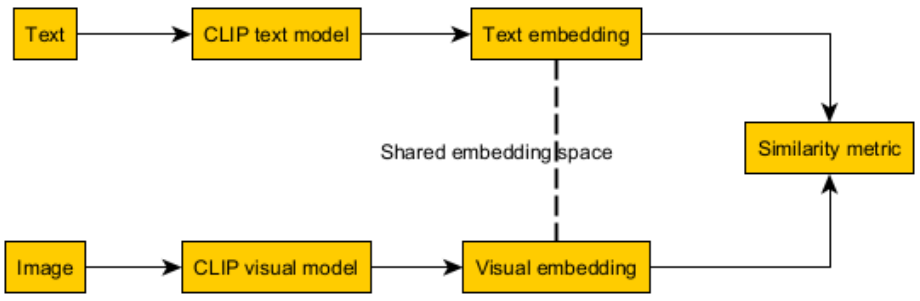
All of the models mentioned in this chapter have been trained with Optuna for HPO (Hyper Parameter Optimization) and evaluated using k-Fold CV (Cross Validation) with 5 folds, unless otherwise noted. Implementation details are available in code for most of the classifiers, unless otherwise noted.

4.2.1 - k-Nearest Neighbors

Figure 4 outlines the label selection method using k-NNs. This method was proposed in (Radford et al., 2021) and served as our first attempted label selection method. The experiments for the k-NN model are available in Appendix A under "./notebooks/eda_embeddings.ipynb".

Figure 4

Label selection using CLIP with nearest-neighbor search



Note. The clip model consists of two components, the visual model and the text model. They convert the input to an embedding space that is shared by both models. Semantically similar contents should have embeddings that are close. We calculated the distances between them using either the dot product or the cosine distance.

4.2.2 - Neural Network Classifiers

We tried training a neural network model to predict the classes of FoodSeg103. The model was composed of 2-4 linear layers, and used LayerNorm, LeakyReLU and Dropout. Each hidden layer was tested with sizes ranging between 8-256, dropout on the interval [0.0, 0.9], and trained with a class-weighted BCEWithLogitsLoss.

The code for the neural network label selection models is available in Appendix A, under "./notebooks/eda_embeddings_nn.ipynb".

4.2.3 - Other Classifiers

We attempted using various classifiers to predict the presence (label 1) or absence (label 0) for a selection of classes from FoodSeg103. We prioritized the classes with a large number of training examples. The classifiers we tested were logistic regression, support vector classifiers and XGBoost. However, the code for these classifiers is currently lost due to human error.

4.2.4 - Clustering

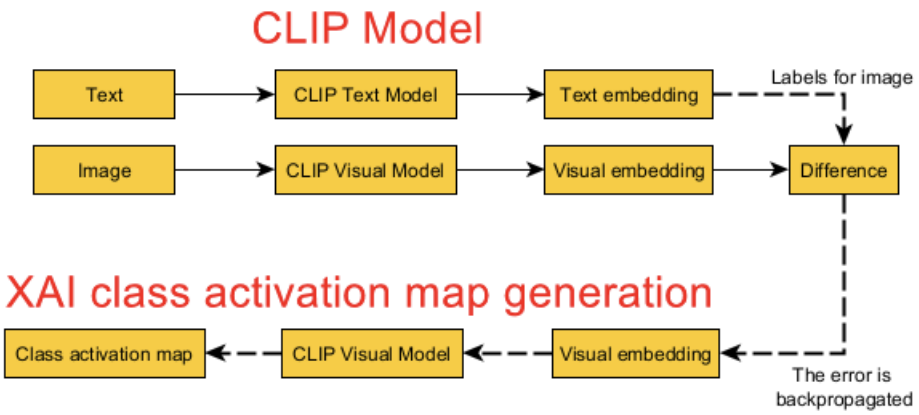
We tested out various clustering methods, including agglomerative clustering, DBSCAN and k-means clustering. Our idea was to see if any of the clustering methods were able to pick up on the patterns from the dataset, and see if cluster metrics aligned with expectations.

The code for the clustering models is available in Appendix A, under "./notebooks/eda_embeddings.ipynb".

4.3 - CLIP With CAM

Figure 5

How to extend the clip model with XAI for generating class activation maps



Note. In the diagram, we have used XAI methods from the CAM family to generate class activation maps. We generate embeddings for the text and image inputs. The text embeddings are then used as the labels for the visual embedding. The difference between them is then backpropagated and used to create a class activation map with the CAM method.

The requirement for the multimodal backbone is that it at least consists of a visual component. We then need to use explainability techniques to generate a mask that explains what parts of the image aligns with the prompt. For this thesis, we've limited ourselves to convolutional networks and text prompts only.

We tested a variety of visual models. From the ResNet family, we tested out both RN50 and RN50X64 (He et al., 2016; Radford et al., 2021). The weights for both these models were the OpenAI-provided weights from OpenCLIP. We also tested the ConvNeXt models, as this family of models claimed to compete favorably with transformer models at the time of publication (Z. Liu et al., 2022). We did consider testing out transformer models as well, but for now we leave that as further work.

For the explainability methods, the requirement is they can look at the visual component of the CLIP model and compare the results with the embeddings from the other modality. They then need to generate a map of what the visual model looked at, when prompted by a specific embedding from the other modality. It must generate a rough segmentation mask that we can sample from and give to the segmentation model in stage 6.

We have focused on class activation mapping methods. We use Grad-CAM, Grad-CAM++ and LayerCAM because they strike a balance between being easy to implement, easy to understand, and have good performance according to (P. Chen et al., 2022).

(P. Chen et al., 2022) benchmarked various explainability methods, showing that the chosen explainability methods have a relatively low impact on speed. The other methods they benchmarked are from one to three orders of magnitude slower. RISE was the worst offender, though, being 793 times slower than Grad-CAM and 203 times slower than LayerCAM. Therefore we chose to focus on the methods above, but aim to make our code as modular as possible to allow for future methods.

The code for CLIP with CAM is available in Appendix A. See the ClipAttentionMapper class under "./segmentation/sscx.py", as well as the code under "./segmentation/xai/".

4.4 - Class Activation Map Filtering

The class activation maps generated by explainability techniques typically have a lot of noise. They generally give some small, but non-zero probability to irrelevant objects and the background, for instance. While many of the CAM methods we applied clamped the output values to a minimum of 0, We didn't know if the maximum value would be bounded to a reasonable scale. Last but not least, we didn't know if the class activation maps that were generated would be a proper probability distribution summing to one. Because of these concerns, we used various filtering techniques to ensure that only regions with high probability were kept.

Through the project, we tried out multiple techniques and ran various experiments. For instance, we tried filtering with thresholding to suppress low probability regions, which we assumed were likely to be noise. We also tried various transformations like squaring or cubing the values to bring low probabilities closer to zero.

Assume we have a class activation map for a particular class $M_c \in \mathbb{R}^{m \times n}$, where m is the mask; c is the class; and $m \times n$ is the size of the input image and the upscaled class activation map. For the operations below, we assume the class activation map is the population when performing distribution operations like calculating the mean and standard deviation. We tried out multiple filtering techniques, some in separation and some in combination. Most often, we

The code for the filters are available in Appendix A. The file "./segmentation/normalizers.py" shows what filtering techniques we implemented. The file "./segmentation/sscx.py", especially the method ClipAttentionMapperConfig.get_normalization_method(), has details on how we configured the filter stacks.

4.4.2 - Identity Function

The identity function is used as part of the filtering framework. It's simply a placeholder that returns the input unchanged.

4.4.2 - Thresholding Functions

The first operation we wish to introduce is thresholding. After performing some filtering, some of the methods require us to remove data, as in the case of the minmax and standard filtering techniques. Thresholding functions will data given some criteria by setting every value of the class activation map to zero.

The basic thresholding technique for some threshold $\rho \in \mathbb{R}$ is:

$$M_c = \begin{cases} M_c & \text{if } M_c > \rho \\ 0 & \text{otherwise} \end{cases}$$

4.4.3 - Percentile Thresholding Filtering

In percentile threshold filtering, we treat the class activation map values as a probability distribution. Let's define the percentile threshold as $p \in (0.0, 1.0)$ and the cutoff value for that threshold as η_p.

Percentile thresholding filtering is then:

$$M_c = \begin{cases} M_c & \text{if } M_c > \eta_p \\ 0 & \text{otherwise} \end{cases}$$

4.4.4 - Standard Normalization Filtering

In this filtering operation, we standardize the data so it has mean of zero and standard deviation of 1. We set a threshold based on the standard deviation and a scalar a ρ, and set every value below the threshold to 0:

$$M_c = \begin{cases} 0 & \text{if } M_c < \rho\sigma \\ M_c & \text{otherwise} \end{cases}$$

4.4.5 - Minmax Normalization Filtering

We used many minmax normalization to make sure the data was scaled between 0 and 1. As previously mentioned, we were worried about values becoming too large or in some methods, maybe too small. Minmax normalization would fix this. We additionally set a threshold to cut off some of the values when using this kind of filtering. The level is a tunable hyperparameter, but we tested with a static value of 0.5.

4.4.6 - Space Transformation Filtering

What we in this paper call space transformation filtering is a transformation from linear space to polynomial space. The idea is to apply a polynomial function to each element of the class activation map. The formula we used is:

$$f(x) = x^p, p \in R, p > 0$$

Other transformations might be useful, but were not explored in this thesis.

4.4.7 - Softmax Filtering

The softmax operation can be applied to push large values to 1 and small values to 0, depending on the distribution of the class activation map.

4.4.8 - Uniform Filtering

Uniform filtering is simply an operation that sets a uniform probability for the whole class map. This is mostly used as a random baseline to compare against.

4.5 - Segmentation Generation

For segmentation generation, we used Segment Anything Model with its point prompting abilities.

The purpose of using the explainability techniques above is to generate a class activation map that gives weight to the parts of the input image that are likely to contain the element we want to segment. However, while explainability techniques can generate segmentation maps, they depend on the underlying model on which they try to explain. The idea then would be to use the rough segmentation masks generated by the explainability technique as input for a model that is specifically meant for segmentation. That means we can combine the strengths of a model that interprets text and images together, but does not create segmentation masks, with the strengths of a model that can not interpret text but is good at creating segmentation masks.

We considered using DINO instead of the Segment Anything Model. The original DINO model would create segmentation masks for a whole image at a time. However, you could not query it by text. Our approach would allow you to sample points from the explainability generated class activation maps, allowing you to probabilistically choose the segmentation masks from DINO that best fit the text query.

4.5.1 - Transforming Class Activation Maps to Probabilities

Before sampling points for the segmentation model, the class activation map must be transformed to a proper probability map. The requirements for this is that the distribution sums to one. There are also other requirements for being a proper probability distribution that we will not go into in this thesis (Devore et al., 2021).

For this subsection, we need some mathematical notation. Let's assume M is the incoming class activation map, that Q is an intermediary and adjusted class activation map, and that T will be the probability mask for the image. $M, T, Q \in R^{m \times n}$. Pixels will be indexable by the variables $x, y \in \mathbb{N}, x \leq n, y \leq m$. Since we are treating the images as matrices in these mathematical operations, they are indexed from one and up.

There are multiple ways to ensure that the class activation map sums to one. The simplest among them, given the nature of the class activation map, would be to divide every element in the class activation map by the sum of the class activation map. However, given that the class activation map might contain non zero values, this might not work. Because of this, we simply choose to subtract the minimum value in the class activation map. There is however, a small chance that this will set all the values to zero. If this occurs, we set equal probability to each pixel. Hence:

$$Q_{x,y} = \begin{cases} M_{y,x} - \min M & \text{if } \sum_{i=1}^{m} \sum_{j=1}^{n} [M_{i,j} - \min M] > 0 \\ \frac{1}{mn} & \text{otherwise} \end{cases}$$

There is a chance that the interpretation of the class activation map changes when you subtract the minimum, but we just want probabilities at this stage and do not consider that.

We can now calculate the final probability map T as:

$$T = p(x, y) = \frac{Q_{y,x}}{\sum_{i=1}^{m} \sum_{j=1}^{n} Q_{i,j}} \; ; \; x, y \in \mathbb{N}, M \in \mathbb{R}^{m \times n}$$

We chose to include this as part of the filtering framework. See Appendix A for the code, which is available in the file "./segmentation/normalizers.py". The class called ToProbabilities contains our implementation.

4.5.2 - Segmenting with Segment Anything Model

In the stages, we generated the class activation mask. We filtered the probability masks, and we created a probability distribution. Now it is time to use that probability distribution with the Segment Anything Model.

From here on the basic idea is to sample a set number of points for the model. This

will give us points in the Cartesian plane that we can feed to the Segment Anything Model. Based on this, the model will generate masks that are appropriate for the input image and the prompted points.

4.6 - Hyperparameter Impact Study Setup

In the hyperparameter impact studies, we test stages 2-6 in an end-to-end fashion. We employ Taguchi designs as our experimental design framework to data-efficiently test multiple variables at the same time. Given that we didn't get the label selection stage to work as well as we had hoped, we decided to test the remaining stage with ground-truth labels instead.

Our null hypothesis is that the parameters are equal, and thus the alternative hypothesis is that they have different effects on the metrics. Since we don't a priori have a preference for either, we will use two-tailed tests to check if there is a difference, and make a choice from there. We run 50,000 repetitions, chosen purely because the tests take a long time to run in Python. This means that the lowest p-value we can see is 2.00*10^-5, and we will thus denote every value below this threshold as ~0.0 - roughly equal to 0.0.

The code for generating data for the studies can be found in Appendix A, in the file "./segmentation/run_taguchi_study.py". We used an L16B orthogonal array for the experiments, which allows for five parameters with four different values each (*L16b Orthogonal Array*, n.d.). In total, the L16B orthogonal array has 16 rows, and we repeated each row 5 times per experiment to gather information about the variance in the resulting studies.

The parameters we tested and their values can be seen in Table 1. They're also available in the file mentioned earlier in the file "./segmentation/run_taguchi_study.py", though the order may have changed between experiments. Additionally, for SAM, we chose to focus on the smallest model provided - ViT-B. The reason for this is that we observed only a small improvement when using ViT-L and ViT-H, but they ran considerably slower. To maximize the number of experiments we could run, we therefore focused mostly on ViT-B.

The results of the studies are available in Appendix A, with separate folders for each dataset and their splits, in the folder "./studies/taguchi/". These files also contain the specific configurations created when running the trials.

Table 1

Available system parameters and values in hyperparameter impact studies

Filtering methods	***XAI Methods***	***CLIP Configurations***	***Polynomial Space Transformation Norms***	***Number of Sampled Points***
none	*uniform*	*Model: convnext_xxlarge* *Weights: laion2b_s34b_b82k_augreg*	*1*	*1*
standard	*gradcam*	*Model: convnext_base_w* *Weights: laion2b_s13b_b82k_augreg*	*2*	*3*
softmax	*gradcampp*	*Model: RN101* *Weights: openai*	*3*	*9*
percentile	*layercam*	*Model: RN50* *Weights: openai*	*4*	*16*

Note. A table of the parameters used in the Taguchi studies.

The file "./notebooks/eda_taguchi.ipynb" was used for data analysis. We use three methods for analysis: SNR (signal-to-noise ratios), permutation testing and bootstrapped confidence intervals. The results are available in Appendix A, in the folders located under "./studies/taguchi/".

4.7 - Summary

In this methodology chapter, we've outlined the framework for our approach. Starting with a system overview, each component has been examined in detail. The process for label selection and the various methods intended for verification have been presented. Our vision for utilizing XAI with CLIP to generate rough class activation maps has been discussed. Multiple modules designed to work together for filtering have been suggested. Furthermore, the integration of these components has been demonstrated by explaining the conversion of class activation maps into probability maps and their subsequent use in guiding SAM for segmentation. Finally, our plan for a study to assess the impact of each hyperparameter within the system has been described.

The next chapter will put this system to the test and present the final results.

5 - Experiments and Results

In this chapter, we present our results.

We will show that our attempts at classification were ultimately not fruitful. We attempted to classify food ingredients from the FoodSeg103 dataset. Our attempts show that some training and fine-tuning on larger food-specific datasets might be necessary.

However, we do find that XAI methods seem to perform decently when guided, and might have a place in human-in-the-loop tasks. As such, all the results we present that evaluate the XAI methods use the ground truth labels from the corresponding dataset as inputs. Since the system is modular, swapping out the classification model for another one is possible, and we evaluate our methods relative to a perfect classifier.

Additionally, we will present the results of extensive system parameter studies on the various parts of our proposed method.

Finally, we will compare our method to state of the art results.

5.1 - Label Selection Results

Overall, the label selection stage of our model has been a failure. It seems that the methods we applied were not enough to classify images into the correct categories.

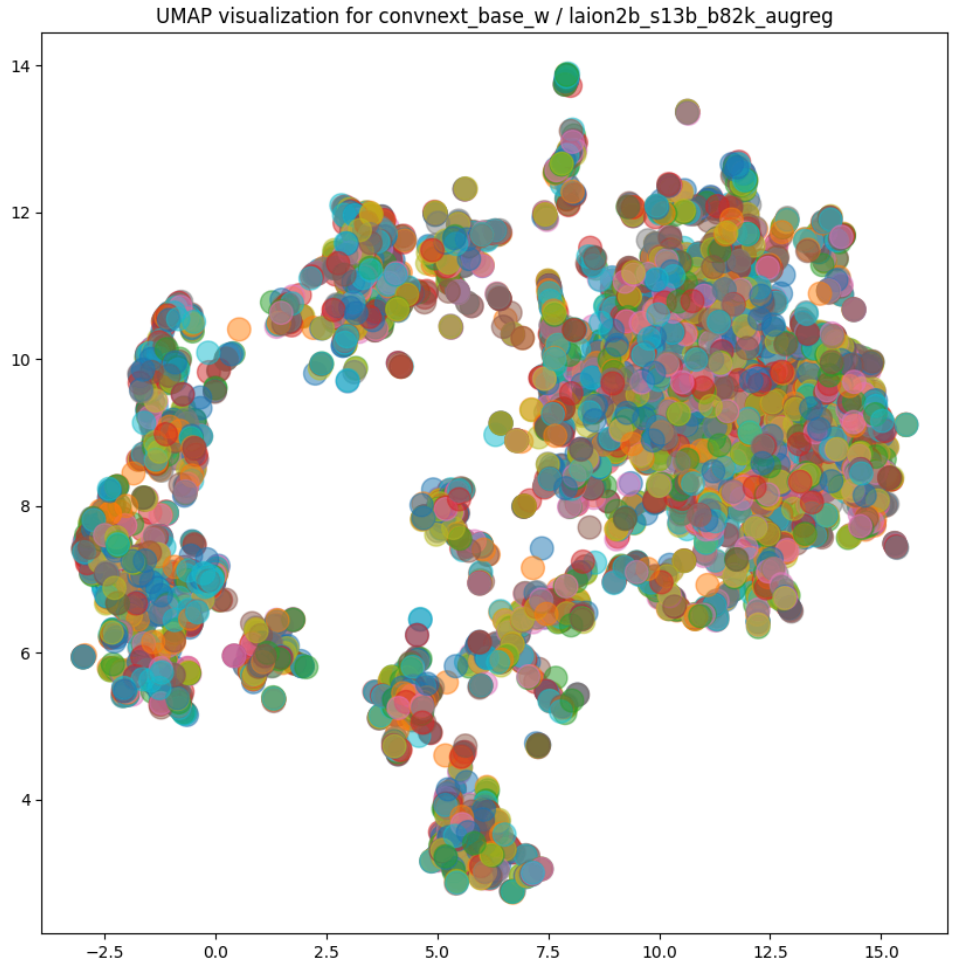
This is likely due to the size of the text and image embedding vectors compared to how much data we have available.

5.1.1 - Dataset Visualization

As part of checking if the embedding space contains relevant information, we tried using UMAP for dimensionality reduction. This might uncover structures in the dataset and hint at whether we can use machine learning techniques for classification. See Figure 6 for the results.

Figure 6

Visualizing the FoodSeg103 dataset training data embeddings with UMAP



Note. A UMAP visualization for the embeddings of the FoodSeg103 dataset, specifically the training subset. The embeddings were generated by OpenCLIP's ConvNeXt Base W model with the weights tag *laion2b_s13b_b82k_augreg.* The plot show little correspondence between ground truth labels and the discovered structure.

5.1.2 - k-NN Model With Visual Embeddings

Our original idea of using nearest neighbor search to avoid having to train a model did not bear fruit. We were inspired by (Radford et al., 2021) and our previous experience with CLIP, and hoped it would work. The code for generating these results is available in Appendix A, in the file "./notebooks/eda_embeddings.ipynb".

After running a hyperparameter search with Optuna for the k-NN model, we got the results in Table 2. These results were achieved with a k-NN model using the following the "ViT-H-14" architecture from OpenCLIP, with "laion2b_s32b_b79k" as the tag for the weights. Optuna found that 58 neighbors was optimal and that 9.36% of neighbors should agree on a class to suggest it.

We tried various methods for deciding how the neighbors should vote, including weighting by rank and distance, but this seemed to have little effect on the results.

The results do not seem too good. Worst case, they are picking up on the class imbalances from the underlying dataset.

Table 2

k-NN model performance for label selection on FoodSeg103 with image embeddings only

Metric name	Validation results	Test results
Accuracy	0.8673	0.8775
F1 score	0.1930	0.2256
Precision	0.1368	0.1598
Recall	0.3509	0.4099

Note. This k-NN model uses visual embeddings only. Validation results are averaged across 5 cross-validation folds.

5.1.3 - k-NN Model With Text/Image Embeddings

The other test we did was to go through the dataset and calculate the distance between the embeddings for the text representations of the labels and the embeddings for the images. That way, we'd classify the images purely based on the labels present in the images.

FoodSeg103

Our best results for FoodSeg103 are available in Table 3. The code for this is available in Appendix A, in the file named "./notebooks/eda_knn_text_image_foodseg103.ipynb".

Again, the results were generally not good. We searched a smaller selection of neighbor values this time and got the best results with OpenCLIP's ConvNeXt XXLarge model with the tag *laion2b_s34b_b82k_augreg* for the weights and 10 neighbors. For this evaluation, we used the text template "Among other ingredients, this dish contains {text}.". Other text templates did not seem to make a difference.

Table 3

k-NN model performance for label selection on FoodSeg103 with both text and image embeddings

Metric name	Train results	Test results
Accuracy	0.8802	0.9114
F1 score	0.0978	0.3295
Precision	0.0669	0.2242
Recall	0.1820	0.6218

Note. This k-NN model uses text embeddings based on to classify the contents of an image. A hyperparameter grid search was performed with the F1 score as the target metric.

UECFoodPix Complete

Our best results for UECFoodPix Complete are available in Table 4. The code for this is available in Appendix A, in the file named "./notebooks/eda_knn_text_image_uecfoodpixcomplete.ipynb".

We used the same setup as with FoodSeg103, only we changed the text template to "This image contains a dish called {text}." The best results were achieved with OpenCLIP's ViT-B-32 model with the weight tag *laion2b_s34b_b79k* and 49 neighbors.

This time the results were surprisingly bad. We have no hypothesis as to what might be the cause given that we've used the same code for both.

Table 4

k-NN model performance for label selection on UECFoodPix Complete with both text and image embeddings

Metric name	Train results	Test results
Accuracy	0.2415	0.2381
F1 score	0.0252	0.0253
Precision	0.0130	0.0131
Recall	0.4327	0.4025

Note. This k-NN model uses text embeddings based on to classify the contents of an image. A hyperparameter grid search was performed with the F1 score as the target metric.

5.1.4 - Neural Networks and Other Classifiers

We were not able to train a satisfying neural network or other machine learning models to function as a label selector.

5.1.5 - Clustering

When evaluated using the Silhouette score, the clustering methods were not able to separate the data into meaningful clusters.

5.1.6 - Summary

Overall, we were unable to train a satisfying label selection component. It's likely that we had too little data in the FoodSeg103 dataset relative to the number of dimensions in the text and visual embeddings generated by CLIP. It could also be that the embeddings generated by CLIP do not contain rich enough information for FoodSeg103 specifically.

5.2 - Evaluation on UECFoodPix Complete

For UECFoodPix Complete, we tested our method on two splits of the dataset: a subset of the train split, and the test split. We intend to show how well our method performs on these splits, how the results compare to other methods, and to analyze how the choice of parameters impact the results.

We ran two experiments: one on the subset of the train split and one on the test split. The purpose of running two experiments was to compare results between the splits, to estimate how well Taguchi methods with different trial configurations could pick up on the variance in the datasets.

We chose a randomly sampled subset of the training data with the same size as the test split, to ensure they both contained 1,000 elements. At the time, we assumed the sampled train split would sufficiently model the distribution of the original train split. While we do not explore this, it's possible that we've underestimated how imbalanced the classes were.

Given that the label selection results were not successful, we chose to use ground-truth classification labels instead. This reduces the noise introduced by a suboptimal label selection module.

5.2.1 - Performance Evaluation

As part of the Taguchi studies, we evaluated the method on a variety of parameter settings. As noted in the methodology section, we tested this with ground-truth labels to decide what classes we should try segmenting.

The results are available in Table 5 for the train split and in Table 6 for the test split. The tables were created with the code in Appendix A, in the file "./notebooks/create_taguchi_tables.ipynb".

For the CLIP config parameter, we used the OpenCLIP weights tag *laion2b_s34b_b82k_augreg* for the convnext_xxlarge model, *laion2b_s13b_b82k_augreg* for the convnext_base_w, and *openai* for both RN50 and RN101.

Table 5

Performance on the UECFoodPix Complete train split, subset of 1,000 samples

CLIP Config	Filtering	XAI Method	Norm	#Samples	aAcc ± 95% CI	mIoU ± 95% CI
convnext_xxlarge	none	gradcampp	2	9	0.8301 ± 0.0177	0.6062 ± 0.0323
convnext_xxlarge	percentile	layercam	1	3	0.8449 ± 0.0096	0.5780 ± 0.0303
convnext_base_w	percentile	gradcam	2	16	0.8144 ± 0.0088	0.5623 ± 0.0185
convnext_base_w	standard	layercam	3	9	0.8105 ± 0.0131	0.5560 ± 0.0173
RN101	standard	gradcampp	1	16	0.7136 ± 0.0187	0.4781 ± 0.0119
convnext_xxlarge	standard	gradcam	4	1	0.8101 ± 0.0106	0.4534 ± 0.0262
RN50	none	layercam	4	16	0.5628 ± 0.0218	0.3767 ± 0.0244
RN101	none	gradcam	3	3	0.7360 ± 0.0252	0.3720 ± 0.0400
RN50	softmax	gradcam	1	9	0.5028 ± 0.0235	0.3189 ± 0.0175
convnext_base_w	softmax	gradcampp	4	3	0.6081 ± 0.0227	0.3075 ± 0.0327
RN101	percentile	uniform	4	9	0.4779 ± 0.0179	0.2974 ± 0.0171
convnext_xxlarge	softmax	uniform	3	16	0.4357 ± 0.0224	0.2927 ± 0.0122
RN50	standard	uniform	2	3	0.5826 ± 0.0315	0.2722 ± 0.0164
RN50	percentile	gradcampp	3	1	0.6681 ± 0.0212	0.2027 ± 0.0046
RN101	softmax	layercam	2	1	0.6626 ± 0.0352	0.2012 ± 0.0506

convnext_base_w	none	uniform	1	1	0.6420 ± 0.0183	0.1750 ± 0.0531

Note. 5 samples were collected for each trial.

From both Table 5, we can see that convnext_xxlarge with no filtering, Grad-CAM++ as the XAI method, 9 sampled points and a polynomial space transformation norm of 2 did quite well on the training data. We additionally tested this configuration on the test dataset where we got an mIoU of 0.58 and an aAcc of 0.80.

Table 6

Performance on the UECFoodPix Complete test split

CLIP Config	**Filtering**	**XAI Method**	**Norm**	**#Samples**	**aAcc ± 95% CI**	**mIoU ± 95% CI**
convnext_xxlarge	percentile	layercam	4	16	0.8076 ± 0.0023	0.5588 ± 0.0064
convnext_xxlarge	none	gradcam	2	3	0.7928 ± 0.0051	0.5416 ± 0.0107
convnext_base_w	standard	gradcam	3	16	0.7622 ± 0.0039	0.5271 ± 0.0036
RN101	standard	layercam	2	9	0.7433 ± 0.0080	0.5099 ± 0.0128
convnext_xxlarge	softmax	gradcampp	3	9	0.6358 ± 0.0069	0.4546 ± 0.0111
RN50	percentile	gradcam	1	9	0.5706 ± 0.0104	0.3974 ± 0.0071
RN101	none	gradcampp	1	16	0.5323 ± 0.0086	0.3932 ± 0.0086
convnext_base_w	none	uniform	4	9	0.5224 ± 0.0087	0.3624 ± 0.0071
RN50	softmax	uniform	2	16	0.4791 ± 0.0120	0.3508 ± 0.0097
convnext_base_w	softmax	layercam	1	3	0.6138 ± 0.0123	0.3406 ± 0.0171
RN50	standard	gradcampp	4	3	0.6579 ± 0.0174	0.3348 ± 0.0226
convnext_base_w	percentile	gradcampp	2	1	0.7093 ± 0.0059	0.3193 ± 0.0124
RN101	percentile	uniform	3	3	0.5914 ± 0.0091	0.3132 ± 0.0121
RN50	none	layercam	3	1	0.6312 ± 0.0123	0.1957 ± 0.0188
RN101	softmax	gradcam	4	1	0.6114 ± 0.0118	0.1762 ± 0.0113
convnext_xxlarge	standard	uniform	1	1	0.6000 ± 0.0137	0.1607 ± 0.0258

Note. 5 samples were collected for each trial.

From Table 5 and Table 6, we see that the aAcc and mIoU results are fairly close. We used separate parameter configurations to see if this would cause considerable changes in the results, but they ended up being fairly close. Differences might be explained by the random sampling of the 1,000 elements we selected as the training subset. We could have tried to preserve the class balances better than we did, but are overall happy with the results.

5.2.2 - SNR Analysis

Table 7 shows signal-to-noise-ratio results from the test and train splits of UECFoodPixComplete. We used the data presented in Table 5 and Table 6 for this analysis.

Table 7

SNR analysis of parameters on UECFoodPix Complete

	SNR for mIoU		**SNR for aAcc**	
Split	**Train**	**Test**	**Train**	**Test**
Number of Sampled Points	**11.67**	**15.73**	2.40	1.48
XAI Method	10.06	5.73	**5.94**	**4.81**
CLIP Config	9.74	4.40	4.26	3.83
Filtering method	8.78	4.22	5.65	3.36
Polynomial Space Transformation Norm	2.02	6.47	3.39	2.91

Note. The table shows the tested parameters, with the resulting SNR values for each metric and each dataset split. The highest values for each metric are shown in bold. The table is sorted by the mIoU SNR value for the train split.

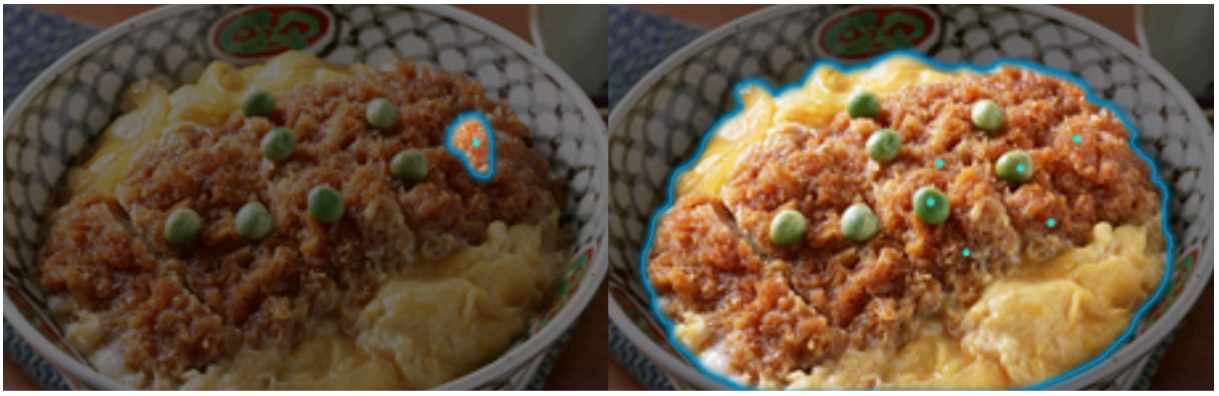
It seems like the number of sampled points has the highest impact on the results. We can clearly see this in Table 5 and Table 6. All the trials with only 1 sampled point are clustered at the bottom of the tables. From informal qualitative evaluations, this makes

sense. With only a few sampled points, SAM might segment out a single pea from a plate of food, as visually shown in Figure 7. It thus makes sense that sampling multiple points of interest could then improve the results. On the other hand, it's possible that multiple points might make the situation worse, if the target object is quite small.

The other thing to note is that uniform filtering performs quite badly. This is expected since uniform filtering should represent random chance, but SAM will get lucky with some set probability.

Figure 7

Visually showing the effect of the number of sampled points



Note. The figure shows two attempted segmentations of a dish from UECFoodPix Complete. On the left, only one point was used as a segmentation reference. On the right, six segmentations point were added in close proximity.

5.2.3 - Hyperparameter impact on mIoU

The analyses performed in this section were generated with the code in Appendix A, in the file "./notebooks/eda_taguchi.ipynb". We run this analysis on the train split only. The full set of data for both the train and test splits can be found in Appendix A, in the folder "./studies/taguchi/UECFoodPixComplete/".

Table 8

UECFoodPix Complete - Parameter analysis - Impact on mIoU from the number of sampled points

Number of Sampled Points	Mean mIoU	95% CI (Bootstrapped)
9	0.4447	[0.3855, 0.5051]
16	0.4274	[0.3827, 0.4723]

3	0.3824	[0.3328, 0.4369]
1	0.2581	[0.2110, 0.3114]

Looking at Table 8, it's clear that sampling a single point will perform worse than the other choices. This is confirmed by permutation tests, which note that
1 point is different from 3 with a p-value of 2.64*10^-3, 9 with a p-value of 1.20*10^-4, and 16 with a p-value of 2.00*10^-5. We observed no other statistically valid results for the number of sampled points.

Table 9
UECFoodPix Complete - Parameter analysis - Impact on mIoU from the XAI Method

XAI Method	**Mean mIoU**	**95% CI (Bootstrapped)**
layercam	0.4280	[0.3595, 0.4927]
gradcam	0.4266	[0.3869, 0.4679]
gradcampp	0.3986	[0.3314, 0.4669]
uniform	0.2593	[0.2355, 0.2802]

Based on the results from Table 9 and the permutation tests, we confirm LayerCAM, Grad-CAM and Grad-CAM++ all perform better than a uniformly distributed class activation map. The p-values when comparing uniform CAMs to other XAI methods are: ~0.0 for gradcam, 6.00*10^-5 for layercam, and 6.80*10^-4 for gradcampp. We found no statistically relevant differences between the other methods.

Table 10
UECFoodPix Complete - Parameter analysis - Impact on mIoU from the CLIP config

CLIP Config	**Mean mIoU**	**95% CI (Bootstrapped)**
convnext_xxlarge	0.4826	[0.4266, 0.5355]
convnext_base_w	0.4002	[0.3271, 0.4716]
RN101	0.3372	[0.2928, 0.3822]
RN50	0.2926	[0.2643, 0.3208]

From Table 10, we can right away see that convnext_xxlarge outperforms the RN variants by a considerable margin. The statistically significant results we found were that convnext_xxlarge and RN50 are different with a p-value of 2.00*10^-5, convnext_xxlarge and RN101 are different with a p-value of 6.40*10^-4, and convnext_base_w is different from RN50 with a p-value of 1.29*10^-2.

Table 11
UECFoodPix Complete - Parameter analysis - Impact on mIoU from the filtering method

Filtering method	**Mean mIoU**	**95% CI (Bootstrapped)**
standard	0.4399	[0.3925, 0.4837]
percentile	0.4101	[0.3382, 0.4810]
none	0.3825	[0.3158, 0.4502]
softmax	0.2801	[0.2577, 0.3001]

From Table 11, we see that softmax filtering is the worst performer and all the other work better than it. The statistically significant results are that all methods are better than softmax, but the other ones seem to perform on par with each other. The relevant p-values are: standard vs softmax with ~0.0, percentile vs softmax with 2.36*10^-3, and finally none vs softmax with 8.00*10^-3.

Table 12
UECFoodPix Complete - Parameter analysis - Impact on mIoU from the choice of polynomial space transformation norm

Polynomial Space Transformation Norm	**Mean mIoU**	**95% CI (Bootstrapped)**
2	0.4105	[0.3330, 0.4880]
1	0.3875	[0.3187, 0.4542]
4	0.3587	[0.3317, 0.3869]
3	0.3558	[0.3004, 0.4141]

As can be surmised from the bootstrapped CI intervals in Table 12, we did not find any statistically significant differences between different values for the polynomial space transformation norms.

5.2.4 - Hyperparameter impact on aAcc

Table 13
UECFoodPix Complete - Parameter analysis - Impact on aAcc from the number of sampled points

Number of Sampled Points	Mean aAcc	95% CI (Bootstrapped)
1	0.6957	[0.6678, 0.7272]
3	0.6929	[0.6477, 0.7402]
9	0.6553	[0.5859, 0.7251]
16	0.6316	[0.5683, 0.6944]

We did not find any statistically significant differences between the number of sampled points and their effects on aAcc. As seen in Table 13, the CIs overlap quite a bit.

Table 14
UECFoodPix Complete - Parameter analysis - Impact on aAcc *from the XAI Method*

XAI Method	Mean aAcc	95% CI (Bootstrapped)
layercam	0.7202	[0.6693, 0.7699]
gradcam	0.7158	[0.6581, 0.7683]
gradcampp	0.7050	[0.6702, 0.7419]
uniform	0.5345	[0.4988, 0.5708]

As with the mIoU results, we see from Table 14 that uniform CAMs perform the worst in this matchup. All other methods are better than uniform CAMs, with the following p-values: gradcampp with ~0.0, layercam with ~0.0, and gradcam with 2.00*10^-5. Aside from this, we found no statistical differences between the methods.

Table 15
UECFoodPix Complete - Parameter analysis - Impact on aAcc *from the CLIP config*

CLIP Config	Mean aAcc	95% CI (Bootstrapped)
convnext_xxlarge	0.7302	[0.6509, 0.7956]
convnext_base_w	0.7187	[0.6771, 0.7593]
RN101	0.6475	[0.6005, 0.6900]
RN50	0.5791	[0.5534, 0.6062]

Yet again, it seems the choice of CLIP model has an impact on the metrics. In this case, convnext_xxlarge performed great again. We found statistically significant differences between the following models: convnext_base_w vs RN50 with p-value ~0.0, convnext_xxlarge vs RN50 with p-value 1.26*10^-3, RN101 vs RN50 with p-value 1.78*10^-2, and finally RN101 vs convnext_base_w with p-value 3.05*10^-2.

Table 16
UECFoodPix Complete - Parameter analysis - Impact on aAcc *from the filtering method*

Filtering method	Mean aAcc	95% CI (Bootstrapped)
standard	0.7292	[0.6860, 0.7686]
percentile	0.7013	[0.6358, 0.7629]
none	0.6927	[0.6489, 0.7366]
softmax	0.5523	[0.5132, 0.5914]

In Table 16, softmax is the loser again, as it performs worse than the other options. We found no statistically significant differences between the other methods. For softmax filtering, the p-values against the others were as follows: uniform vs standard had the p-value ~0.0, uniform vs none had t p-value 8.00*10^-5, and uniform vs percentile had the p-value 7.00*10^-4.

Table 17
UECFoodPix Complete - Parameter analysis - Impact on aAcc *from the choice of polynomial space transformation norm*

Polynomial Space Transformation Norm	Mean aAcc	95% CI (Bootstrapped)

2	0.7224	[0.6756, 0.7674]
1	0.6758	[0.6219, 0.7302]
3	0.6626	[0.5985, 0.7214]
4	0.6147	[0.5638, 0.6703]

In Table 17, we see a single statistically significant value, with the norm of 2 being different from 4 with p-value 6.86*10^-3.

5.2.6 - Test Results and Comparison Against State of the Art

Using the results from the subsection analyzing how parameters impact mIoU, we evaluated the method on the test dataset. We used ConvNeXt XXLarge with the weight tag *laion2b_s34b_b82k_augreg* as the CLIP config, LayerCAM as the XAI method, standard filtering as the filtering method and a polynomial space transformation with a norm of 2. We sampled 9 points from the probability masks. With this configuration, we achieved an mIoU of 0.5879 ±0.0075 (95% bootstrapped CI) and an aAcc of 0.8110 ± 0.0041 (95% bootstrapped CI).

Table 18

Comparing Our Method on UECFoodPix Complete Against State of the Art

Method	**mIoU**	**aAcc**
deeplabV3+ †	0.5550	0.6680
YOLACT †	0.5485	N/A
FCN8 ‡	0.558	N/A
SegNet ‡	0.576	N/A
Our method (With ground-truth classes)	0.5879	0.8110
GourmetNet †	0.6288	0.8707
BayesianDeeplabv3+ †	0.6421	0.8729

deeplabV3+ (FoodSAM baseline) †	0.6561	0.8820
FoodSAM †	0.6614	**0.8847**
-CCNet ‡	0.679	N/A
DANet ‡	0.684	N/A
CANet ‡	**0.689**	N/A

Note. The table combines benchmark results from (Lan et al., 2024) † and (Dong et al., 2023) ‡. We sometimes refer to those papers rather than the sources they reference due to uncertainties with who the specific authors are. E.g. (Dong et al., 2023) mentions FCN8, but references the original article, which does not mention training on UECFoodPix Complete.

Our results do not beat the state of the art. When paired with the fact that we currently do not have a functioning label selector for UECFoodPix Complete, the results are also somewhat misleading. However, we show that the method performs within expectation as a guide for SAM, and compete with supervised methods when we know what labels to segment on.

5.2.7 - Summary

In this section, we showed how well our method worked on a subset of the train split and the test split from the UECFoodPixComplete dataset. We used a Taguchi experimental design to analyze the data, providing an orthogonal grid search of different parameters. Through the chapter, we provided results on how well our XAI based method can guide another segmentation module like SAM.

In addition to this, we evaluate our method on the UECFoodPix Complete dataset, where we achieved an mIOU of 0.5879 ±0.0075 (95% bootstrapped CI) and an aAcc of 0.8110 ± 0.0041 (95% bootstrapped CI). Compared to the SotA, our results are not that impressive, with other methods reaching an mIoU of 0.689 and an aAcc of 0.806.

5.3 - Evaluation on FoodSeg103

For FoodSeg103, we tested our method on the train split, and additionally evaluated the method on the test dataset. Through FoodSeg103, we intend to show how our method tackles a harder dataset, as FoodSeg103 consists primarily of ingredient-level annotations and is deemed to be more challenging than UECFoodPix Complete (Wu et al., 2021).

We tested the experimental study design methodology on the UECFoodPix Complete

dataset and saw similar results between the train and test splits.Therefore, we believe that the approach should yield similar results here, and do not use the test split until we've decided on the optimal parameter configuration to try.

To decrease the runtime, we sampled a random subset of 2,000 samples from the train split.

5.3.1 - Performance Evaluation

As part of the Taguchi studies, we evaluated the method on a variety of parameter settings. As noted in the methodology section, we tested this with ground-truth labels to decide what classes we should try segmenting.

The results for the train split are available in Table 19. The table was created with the code in Appendix A, in the file "./notebooks/create_taguchi_tables.ipynb".

For the CLIP config parameter, we used the OpenCLIP weights tag *laion2b_s34b_b82k_augreg* for the convnext_xxlarge model, *laion2b_s13b_b82k_augreg* for the convnext_base_w, and *openai* for both RN50 and RN101.

Table 19

Performance on the FoodSeg103 train split, subset of 2,000 samples

CLIP Config	Filtering	XAI Method	Norm	#Samples	aAcc ± 95% CI	mIoU ± 95% CI
convnext_xxlarge	percentile	layercam	4	16	0.8385 ± 0.0057	0.4318 ± 0.0069
convnext_base_w	standard	gradcam	3	16	0.8133 ± 0.0022	0.4268 ± 0.0038
convnext_xxlarge	none	gradcam	2	3	0.8085 ± 0.0053	0.3679 ± 0.0057
convnext_base_w	percentile	gradcampp	2	1	0.8624 ± 0.0073	0.2679 ± 0.0123
RN101	standard	layercam	2	9	0.6631 ± 0.0045	0.2217 ± 0.0047
convnext_xxlarge	softmax	gradcampp	3	9	0.5319 ± 0.0101	0.2154 ± 0.0092
RN50	standard	gradcampp	4	3	0.7416 ± 0.0114	0.1741 ± 0.0059
RN50	percentile	gradcam	1	9	0.4679 ± 0.0146	0.1589 ± 0.0062
RN101	none	gradcampp	1	16	0.3771 ± 0.0061	0.1455 ± 0.0055
convnext_base_w	softmax	layercam	1	3	0.6164 ± 0.0062	0.1452 ± 0.0136

RN50	softmax	uniform	2	16	0.3505 ± 0.0073	0.1296 ± 0.0055
convnext_base_w	none	uniform	4	9	0.4026 ± 0.0082	0.1295 ± 0.0044
RN101	percentile	uniform	3	3	0.5892 ± 0.0115	0.1105 ± 0.0076
RN50	none	layercam	3	1	0.7654 ± 0.0106	0.0851 ± 0.0163
RN101	softmax	gradcam	4	1	0.7396 ± 0.0019	0.0610 ± 0.0085
convnext_xxlarge	standard	uniform	1	1	0.7370 ± 0.0082	0.0598 ± 0.0077

Note. 5 samples were collected for each trial. Norm refers to the polynomial space transformation norm and #samples to the number of sampled points.

From Table 19, we see that the results are quite different from the UECFoodPix Complete Results in Table 5 and Table 6. This hints at the dataset being more challenging. The difference in mIoU might also be from differences in class balancing, but we did not explore this in detail.

5.3.2 - SNR Analysis

Table 20 shows signal-to-noise-ratio results from train subset split of UECFoodPixComplete. We used the data presented in Table 19 for this analysis.

Table 20

SNR analysis of parameters on FoodSeg103, on 2,000 samples from the train split

	SNR for mIoU	**SNR for aAcc**
Number of Sampled Points	**18.78**	**8.45**
Polynomial Space Transformation Norm	13.27	4.45
XAI Method	13.10	7.27
CLIP Config	11.48	5.25
Filtering method	10.56	6.22

Note. The table shows the tested parameters, with the resulting SNR values for each metric. The highest values for each metric are shown in bold. The table is sorted by the mIoU SNR value.

Again, the number of sampled points is the parameter with the highest impact. Beyond that, there's a difference between the SNR numbers in Table 20 and Table 7. On the train splits, the transformation norm has a higher impact on the performance on FoodSeg103 than on UECFoodPix Complete. However, the order is otherwise the same. We also note that the order is the same between the FoodSeg103 train split and the UECFoodPix Complete test split.

We see the same trends again - sampling only a single point or using uniform filtering both perform badly.

5.3.3 - Hyperparameter impact on mIoU

The analyses performed in this section were generated with the code in Appendix A, in the file "./notebooks/eda_taguchi.ipynb". The full set of data subset of the train split can be found in Appendix A, in the folder "./studies/taguchi/FoodSeg103/".

Table 21
FoodSeg103 - Parameter analysis - Impact on mIoU from the number of sampled points

Number of Sampled Points	Mean mIoU	95% CI (Bootstrapped)
16	0.2834	[0.2232, 0.3434]
3	0.1994	[0.1582, 0.2455]
9	0.1814	[0.1642, 0.1982]
1	0.1184	[0.0835, 0.1588]

Table 22
FoodSeg103 - Parameter analysis - Statistically significant mIoU differences between the number of sampled points

#Samples 1	#Samples 2	P-value	Significant?
1	9	~0.0	Yes
3	9	~0.0	Yes
1	3	9.00*10^-4	Yes

1	16	2.20*10^-3	Yes
3	16	1.05*10^-1	Yes
9	16	1.78*10^-1	

This time, the effects of the number of sampled points parameter was quite a bit stronger. See Table 22 for the results. The only non-significant difference was between 3 and 9 sampled points, with a p-value of 4.6252*10^-1.

Table 23

FoodSeg103 - Parameter analysis - Impact on mIoU from the XAI Method

XAI Method	**Mean mIoU**	**95% CI (Bootstrapped)**
gradcam	0.2537	[0.1879, 0.3186]
layercam	0.2209	[0.1660, 0.2803]
gradcampp	0.2007	[0.1808, 0.2212]
uniform	0.1073	[0.0943, 0.1193]

Based on the results from Table and the permutation tests, we confirm LayerCAM, Grad-CAM and Grad-CAM++ all perform better than a uniformly distributed class activation map. The p-values when comparing uniform CAMs to other XAI methods are: ~0.0 for gradcampp, 1.40*10^-4 for layercam, and 1.80*10^-4 for gradcam. We found no statistically relevant differences between the other methods.

Table 24

FoodSeg103 - Parameter analysis - Impact on mIoU from the CLIP config

CLIP Config	**Mean mIoU**	**95% CI (Bootstrapped)**
convnext_xxlarge	0.2687	[0.2048, 0.3307]
convnext_base_w	0.2424	[0.1921, 0.2964]
RN50	0.1369	[0.1215, 0.1515]
RN101	0.1347	[0.1094, 0.1606]

In the case of Table 24, we immediately see that the ConvNeXt models outperform the ResNet models. We found the following statistical differences between the models.

convnext_base_w beats RN50 with a p-value of 2.00*10^-4. convnext_xxlarge beats RN50 with a p-value of 4.80*10^-4. convnext_xxlarge beats RN101 with a p-value of 7.60*10^-4. convnext_base_w beats RN101 with a p-value of 1.00*10^-3. Beyond this, we did not find significant differences between convnext_xxlarge and convnext_base_w or between RN50 and RN101.

Table 25

FoodSeg103 - Parameter analysis - Impact on mIoU from the filtering method

Filtering method	**Mean mIoU**	**95% CI (Bootstrapped)**
percentile	0.2423	[0.1901, 0.2971]
standard	0.2206	[0.1649, 0.2800]
none	0.1820	[0.1365, 0.2324]
softmax	0.1378	[0.1138, 0.1621]

In Table 25, we see that softmax underperforms once again. Percentile filtering beats softmax filtering with a p-value of 1.46*10^-3, and so does standard filtering with a p-value of 1.62*10^-2. This is in line with the results from UECFoodPix Complete. However, on FoodSeg103, the none option was equally bad, and we found no statistical difference between none and softmax.

Table 26

FoodSeg103 - Parameter analysis - Impact on mIoU from the choice of polynomial space transformation norm

Polynomial Space Transformation Norm	**Mean mIoU**	**95% CI (Bootstrapped)**
2	0.2468	[0.2093, 0.2847]
3	0.2095	[0.1528, 0.2710]
4	0.1991	[0.1412, 0.2631]
1	0.1273	[0.1089, 0.1435]

In Table 26, we can immediately see that not using the norm parameter (i.e. using a norm of 1) gives the worst results. The statistically relevant results we found were as follows:

2 beats 1 with p-value ~0.0, 3 beats 1 with p-value 1.44*10^-2, and 4 beats 1 with p-value 3.73*10^-2. We found no statistically significant differences between the other values.

5.3.4 - Hyperparameter impact on aAcc

Table 27

FoodSeg103 - Parameter analysis - Impact on aAcc from the number of sampled points

Number of Sampled Points	Mean aAcc	95% CI (Bootstrapped)
1	0.7761	[0.7549, 0.7998]
3	0.6889	[0.6497, 0.7283]
16	0.5948	[0.4984, 0.6905]
9	0.5164	[0.4747, 0.5592]

Unlike with the UECFoodPix Complete dataset, the number of sample points has a big impact on the aAcc when run on FoodSeg103. We conjecture that this could be because sampling a single point is less likely to lead to large segmentation masks than multiple points, and thus less likely to make large errors affecting mIoU. While mIoU and aAcc seem correlated, it can be a trade-off between them.

We find statistically significant differences between sampling a single point and 9 points with p-value ~0.0, 3 points with p-value 9.00*10^-4, and 16 points with p-value 2.20*10^-3. Additionally, there's a statistically significant difference between sampling 3 and 9 points, with p-value ~0.0.

Table 28

FoodSeg103 - Parameter analysis - Impact on aAcc *from the XAI Method*

XAI Method	Mean aAcc	95% CI (Bootstrapped)
layercam	0.7208	[0.6833, 0.7585]
gradcam	0.7073	[0.6422, 0.7654]
gradcampp	0.6282	[0.5466, 0.7095]
uniform	0.5198	[0.4533, 0.5882]

From looking at the CIs in in table 28, we immediately see that layercam and gradcam perform significantly better than uniform CAMs, with p-values 2.00*10^-05 and 5.00*10^-04 respectively. However, while gradcampp has a considerably higher mean and upper bound, the lower bound overlaps enough with the uniform CAM approach. Thus, we cannot reject the null hypothesis that the impact of gradcamcpp and uniform on aAcc might be the same.

Table 29

F*oodSeg103 - Parameter analysis - Impact on* aAcc *from the CLIP config*

CLIP Config	**Mean aAcc**	**95% CI (Bootstrapped)**
convnext_xxlarge	0.7290	[0.6746, 0.7786]
convnext_base_w	0.6737	[0.5931, 0.7505]
RN101	0.5922	[0.5311, 0.6494]
RN50	0.5813	[0.5028, 0.6578]

Table 29 immediately tells us that convnext_xxlarge is different from both convnext_xxlarge and RN50, with the statistically significant p-values 2.42*10^-3 and 5.20*10^-3 respectively. We fail to reject the null hypothesis for any of the other model pairings.

Table 30

FoodSeg103 - Parameter analysis - Impact on aAcc *from the filtering method*

Filtering method	**Mean aAcc**	**95% CI (Bootstrapped)**
standard	0.7388	[0.7157, 0.7619]
percentile	0.6895	[0.6165, 0.7619]
none	0.5884	[0.5049, 0.6739]
softmax	0.5596	[0.4970, 0.6209]

Again, softmax comes out as the losing filtering method, beaten by both standard filtering with p-value ~0.0, and percentile filtering with p-value 1.44*10^-2. Additionally, we see standard filtering beating none with p-value 3.36*10^-3.

Table 31

FoodSeg103 - Parameter analysis - Impact on aAcc *from the choice of polynomial space transformation norm*

Polynomial Space Transformation Norm	**Mean aAcc**	**95% CI (Bootstrapped)**
4	0.6806	[0.6039, 0.7489]
3	0.6749	[0.6233, 0.7257]
2	0.6711	[0.5808, 0.7538]
1	0.5496	[0.4895, 0.6100]

For FoddSeg103, it seems that a higher norm might be a better choice, with 1 being the worst by far. A norm of 3 beats it with p-value 5.10*10^-3, a norm of 4 with p-value 1.21*10^-2 and a norm of 2 with p-value 3.56*10^-2. However, we fail to reject that norms 2-4 could have the same impact.

5.3.6 - Test Results and Comparison Against State of the Art

Using the results from the subsection analyzing how parameters impact mIoU, we evaluated the method on the test dataset. We used ConvNeXt XXLarge with the weight tag *laion2b_s34b_b82k_augreg* as the CLIP config, Grad-CAM as the XAI method, percentile filtering as the filtering method and a polynomial space transformation with a norm of 2. We sampled 16 points from the probability masks. With this configuration, we achieved an mIoU of 0.4178 ± 0.0055 (95% bootstrapped CI) and an aAcc of 0.8096 ± 0.0033 (95% bootstrapped CI).

Table 32

Comparing Our Method on FoodSeg103 Against State of the Art

Method	**mIoU**	**aAcc**
BEiT v2 Large *	**0.494**	N/A
FoodSAM †	0.4642	0.8410
SeTR-MLA *	0.451	0.8353

SeTR-Naive *†	0.439	N/A
Our method (With ground-truth classes)	0.4178	0.8096
Swin-S *	0.416	N/A
CCNet-Finetune †	0.413	**0.877**
InternImage-B *	0.411	N/A
STPPN †	0.403	0.8213
Upernet †	0.398	0.8202
CANet (ReLeM) ‡	0.3721	N/A
CCNet (ReLeM) ‡	0.3665	N/A
CCNet *	0.355	N/A
DANet (ReLeM) ‡	0.3544	N/A
CANet ‡	0.3534	N/A
DANet ‡	0.3472	N/A
CCNet ‡	0.3418	N/A
Window Attention †	0.314	0.7762
ReLeM-FPN-Finetune †	0.308	0.789
ReLeM-CCNet †	0.292	0.793
CCNet †	0.286	0.789
FPN †	0.2728	0.7523

Note. The table combines benchmark results from (Lan et al., 2024) †, (Dong et al., 2023) ‡ and (Sinha et al., 2023) *. We sometimes refer to those papers rather than the sources they reference due to uncertainties with who the specific authors are. E.g. (Dong et al., 2023) mentions FCN8, but references the original article, which does not mention training on UECFoodPix Complete.

Again, our method does not beat the state of the art, and we still do not have a functioning label selector for FoodSeg103 either. As mentioned before, these results are somewhat misleading. However, they do show that XAI can guide SAM to competitive segmentation results, if a sufficient label selector was available.

5.3.7 - Summary

In this section, we showed that our method can create reasonable segmentation suggestions for the FoodSeg103 dataset. We use Taguchi experimental design to analyze the data, in which we performed a grid search with an orthogonal array. The chapter provides information on how the various parameters impact metrics.

We also evaluated the parameters found in the chapter on the FoodSeg103 test split, on which we achieved an mIoU of 0.4178 ± 0.0055 (95% bootstrapped CI) and an aAcc of 0.8096 ± 0.0033 (95% bootstrapped CI). When compared against the SotA, our results lag behind, with the best method we found scoring an mIoU of 0.494.

5.4 - Summary

Overall, the results have positive and negative sides. The positives include how well our method seems to perform under optimal conditions. When guided by a strong classifier, we see competitive performance on the benchmarks for both UECFoodPix Complete and FoodSeg103. While the performance is not state of the art, they would likely be better if we had a stronger multimodal model. Additionally, they would probably perform better if both the multimodal model and the segmentation model were fine-tuned on food images. On the negative side, the label selection components did not seem to work well. We speculate that this could be because the data is too high-dimensional, but it's also possible that the model is not good at picking up food.

We will discuss the results further in the next chapter and consider whether we've sufficiently answered our research questions.

6 - Discussion

In the discussion chapter, we will talk about how well our proposed method did. We'll look at our original problem statement and research questions, and discuss our answers to those questions.

6.5 - Food Image Datasets

Overall, it would seem there are some challenges in food segmentation datasets. Most of the datasets have up to three types of labels: presence of food, different dishes or different ingredients.

The datasets come in different levels of difficulty. For instance, if they have a mix of different dishes or ingredients that could make it difficult to separate the contents within them, as mentioned in (Y. Wang et al., 2019) and (Wu et al., 2021). Other problems include the homogeneity of some of the datasets, especially if the images are captured using the same type of equipment in the same location.

Furthermore, it is difficult to compare the results from different datasets across each other. Some of them suggest using precision, recall and F1 scores as the metrics. Others report mIoU, aAcc and mAcc. Some even introduce cIoU. Of course, this doesn't have to be a problem. Most of the time, one is interested in training a model and evaluating on specific datasets, not comparing them against each other. Thus, it makes sense to use metrics that the dataset and relevant tasks have converged on. To get an idea of the complexity of a particular dataset, you would find a paper where models are trained on different datasets. and evaluate them.

It seems that several datasets have used automatically generated segmentation masks, as is the case in UEC-FooxPix, SUEC Food and Food201-Segmented (Ege et al., 2019; Gao et al., 2019; Meyers et al., 2015; Okamoto & Yanai, 2021). When you train a model on such a dataset, you rely on the original segmentation model being good enough. If you don't need to train on the dataset itself, you still need to rely on it for evaluation. This introduces some level of uncertainty to these datasets.

Another huge challenge with these datasets is that they contain imbalanced data. Some classes might not be well represented. For instance, FoodSeg103 has 103 ingredients, 7,118 images in total and 42,097 class occurrences. Within this distribution, the pudding class occurs only 6 times - 5 in training and 1 in testing. For comparison, the ingredient most often seen is steak, with 2,436 occurrences in total.

6.5 - Adapting to New Segmentation Models

The only method we have tested out so far is the segment anything model. However, adapting to new segmentation models should be relatively easy if it's possible to use the class activation maps to generate 2D maps in some way.

As previously speculated, you might use the method with the segmentation masks that DINO creates (Caron et al., 2021). As we understand it, DINO practically performs instance segmentation. Thus it would be relatively forward to take the segmentation masks from DINO and use the class activation map to probabilistically select the segmentation masks that are most relevant. You could do this for instance, by sampling points from the probability space and see where they overlap with the segmentation maps from DINO. You could also weigh the full segmentation space from DINO with the class annotation maps, generate a center of mass probability for each of the masks from DINO and then select those that are above some threshold.

6.6 - Adapting the Method to New Modalities

Adapting the method to a new modality like audio or 3D data would be relatively easy, as mentioned previously. One would just need a new encoder that is trained in a multimodal fashion by swapping out the text encoder for the new modality. That still means you need to find an appropriate encoder with pretrained weights, or possibly train one yourself. If it doesn't already exist, this would be a promising research direction. You might not have the resources to train a model yourself from scratch, because both the computational costs and gathering of data might be prohibitively expensive. However, adapting an existing encoder using the CLIP-inspired fine-tuning should be relatively forward.

6.7 - Criticism

The biggest criticism of our proposed method is probably - by don't we just train? It should be too expensive to train a model in latent space, if we wanted to create a new prompt encoder for SAM. At least, the hope is that this would be possible. For instance, if SAM could be trained to accept the embeddings from the *omnimodal* ImageBind, haven't we almost solved the problem already? Well, that might be true.

However, we'd like to point out that ideas come in unconventional shapes and forms, and XAI guidance seemed like an idea that begged to be explored.

7 - Conclusion

7.1 - Thesis Summary

In this thesis, we explored the use of explainable AI methods as a guide for food image segmentation. We used class activation mapping methods to generate maps of interest in images that we tried to refine through filtering. Based on the filtered maps, we sampled points that we sent to a general-purpose segmentation model called Segment Anything Model.

We ran an extensive set of tests to determine what hyperparameters had a high level of impact on the system. Using statistical methods, we determined that the number of sampled points, the multimodal model used, and filtering techniques seemed to have an impact on the result. We concluded that all XAI methods performed considerably better than random guessing.

7.2 - Main Contributions

7.2 - Revisiting the Problem Statement

7.2.1 - Can we adapt a general-purpose multimodal model for food image classification with minimal training?

The short answer is no, we were not able to adapt our CLIP for food image classification. We tried to do this in multiple ways, including k-NN search,

The problem seems to be that the CLIP embeddings are that this just doesn't work for FoodSeg103. After using PCA to reduce the number of features down to a single dimension, we got similar results. This is likely due to a combination of two factors.

First, the embeddings generated by CLIP are generally large. For the model architectures we tested, they range from 512-1024 dimensions. Given the small number of training examples and the high number of dimensions, any model is likely to overfit. The alternative is that the signal provided by CLIP isn't strong enough to rank relevant text labels at the top, if the image contains many competing or small details.

We think these results could be solved if we were to choose a stronger multimodal model like SigLiT. Additionally, it can be fine-tuned for food, for instance using the Recipe1M or Recipe1M+ datasets (Marin et al., 2019; Salvador et al., 2017). This means that the method would need to be trained, which would add additional resource requirements to the method. However, this would be a promising way to go for future work.

7.3.2 - Are the segmentation maps created by explainability techniques good enough to guide general-purpose segmentation models?

Based on the benchmarks we collected for both UECFoodPix Complete and FoodSeg103, it seems that XAI guidance with CAM-based methods performs reasonably well. All the tested XAI methods perform considerably better than our random chance baseline. We also see that the method depends on the model used, with ConvNeXt models performing the best. We conjecture that this is likely due to the size of the dataset used to train that model, as both ResNet variants were trained with OpenAI's dataset instead of Laion5B.

We do not deliver state of the art results, but we also didn't expect to. We just hoped our results would show that the method had potential.

However, with a stronger multimodal model fine-tuned for food, we might be able to deliver even better results than we currently do, which could make XAI guidance a competitive choice. It might also be possible to use a model fine-tuned on food as the segmentation provider, whether that is SAM or DINO. Better understanding of what makes up a single dish or ingredient across the full image would be helpful.

7.3.3 - Can we improve on the maps generated by XAI?

The answer to this seems to be somewhat mixed.

We found only a single statistically significant result where using a particular type of filtering was better than using "none". This results was on FoodSeg, where using standard filtering improved the aAcc by a p-value of 3.36*10^-3.

The results look better for the use of the polynomial space transformation norms, though. The trend was that leaving the norm at 1 had a detrimental effect on both mIoU and aAcc for FoodSeg103 and the test split of UECFoodPix Complete. However, we did not see the same patterns for the subset of the UECFoodPix Complete train split. This might be due to a weakness in the Taguchi design we ended up with, the amount of data we collected, or something else.

However, we think refinement of the XAI generated probability maps is fascinating and welcome more research into them.

7.3.4 - Are general purpose segmentation models good enough for food image segmentation?

Based on the results we have observed so far - absolutely! Even though we used ground truth labels since our label selector didn't work, we see a wide spread in how well

stages 2-6 of our method worked. At its best, the segmentation masks that an XAI-guided SAM is able to generate compete favorably with the other methods in our benchmark. Of course, label selection is a huge part of any segmentation model, and we're not downplaying this. What we want to highlight is just that SAM can do well with a good label selector!

We believe SAM might be even better if it was fine-tuned specifically on food data.

7.4 - Future Work

For future work, we'd like to see if our methods might perform better with a different multimodal model or segmentation model. We also see potential in fine-tuning the multimodal model or segmentation model specifically on food, and possibly reducing the embedding size to avoid overfitting.

Our filtering methods are currently underexplored. Our use of Taguchi methods was interesting, but we didn't end up covering a lot of possible techniques or hyperparameters. For instance, the thresholds might benefit from being set to other values.

To get a better overview of how well the method performs, it could be useful to try it on different datasets, both food and other types. In addition to that, further qualitative studies of the XAI methods used with CLIP, as well as the having the method evaluated by humans, might reveal where the failure cases might come from.

8 - References

Addo-Tenkorang, R., & Helo, P. T. (2016). Big data applications in operations/supply-chain management: A literature review. *Computers & Industrial Engineering*, *101*, 528–543. https://doi.org/10.1016/j.cie.2016.09.023

Altin, M., & Cakir, A. (2024). *Exploring the Influence of Dimensionality Reduction on Anomaly Detection Performance in Multivariate Time Series* (arXiv:2403.04429). arXiv. http://arxiv.org/abs/2403.04429

Aslan, S., Ciocca, G., Mazzini, D., & Schettini, R. (2020). Benchmarking algorithms for food localization and semantic segmentation. *International Journal of Machine Learning and Cybernetics*, *11*(12), 2827–2847. https://doi.org/10.1007/s13042-020-01153-z

Aslan, S., Ciocca, G., Mazzini, D., & Schettini, R. (2024). *Benchmarking Algorithms for Food Localization and Semantic Segmentation*. Imaging and Vision Laboratory. http://www.ivl.disco.unimib.it/activities/benchmarking-food-segmentation/

Baltrušaitis, T., Ahuja, C., & Morency, L.-P. (2017). *Multimodal Machine Learning: A Survey and Taxonomy* (arXiv:1705.09406). arXiv. https://doi.org/10.48550/arXiv.1705.09406

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, *58*, 82–115. https://doi.org/10.1016/j.inffus.2019.12.012

Battini Sönmez, E., Memiş, S., Arslan, B., & Batur, O. Z. (2023). The segmented UEC Food-100 dataset with benchmark experiment on food detection. *Multimedia Systems*, *29*(4), 2049–2057. https://doi.org/10.1007/s00530-023-01088-9

Blumenstiel, B., Jakubik, J., Kühne, H., & Vössing, M. (2023). *What a MESS: Multi-Domain Evaluation of Zero-Shot Semantic Segmentation* (arXiv:2306.15521). arXiv. https://doi.org/10.48550/arXiv.2306.15521

Busra Emek Soylu, Mehmet Serdar Güzel, Erkan Bostancı, Fatih Ekinci, Tunç Aşuroğlu, & Koray Açıcı. (2023). *Deep-Learning-Based Approaches for Semantic Segmentation of Natural Scene Images: A Review*. *12*(12), 2730–2730. https://doi.org/10.3390/electronics12122730

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., & Joulin, A. (2021). *Emerging Properties in Self-Supervised Vision Transformers* (arXiv:2104.14294). arXiv. https://doi.org/10.48550/arXiv.2104.14294

Chattopadhyay, A., Sarkar, A., Howlader, P., & Balasubramanian, V. N. (2018). Grad-CAM++: Improved Visual Explanations for Deep Convolutional Networks. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 839–847. https://doi.org/10.1109/WACV.2018.00097

Chen, L.-C., Papandreou, G., Kokkinos, I., Kevin Murphy, Kevin Murphy, Kevin Murphy, Murphy, K., & Yuille, A. L. (2018). DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(4), 834–848. https://doi.org/10.1109/tpami.2017.2699184

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). *Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation*. 833–851. https://doi.org/10.1007/978-3-030-01234-2_49

Chen, M.-Y., Yang, Y.-H., Ho, C.-J., Wang, S.-H., Liu, S.-M., Chang, E., Yeh, C.-H., & Ouhyoung, M. (2012). Automatic Chinese food identification and quantity estimation. *SIGGRAPH*

Asia 2012 Technical Briefs, 1–4. https://doi.org/10.1145/2407746.2407775

Chen, M.-Y., Yang, Y.-H., Ho, C.-J., Wang, S.-H., Liu, S.-M., Yeh, C.-H., & Ouhyoung, M. (n.d.). *Automatic Chinese Food Identification and Quantity Estimation*. Communications and Multimedia Laboratory. Retrieved May 6, 2024, from https://www.cmlab.csie.ntu.edu.tw/project/food/

Chen, P., Li, Q., Biaz, S., Bui, T., & Nguyen, A. (2022). *gScoreCAM: What objects is CLIP looking at?* 1959–1975. https://openaccess.thecvf.com/content/ACCV2022/html/Chen_gScoreCAM_What_objects_is_CLIP_looking_at_ACCV_2022_paper.html

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Chen, W.-G., Spiridonova, I., Yang, J., Gao, J., & Li, C. (2023). *LLaVA-Interactive: An All-in-One Demo for Image Chat, Segmentation, Generation and Editing* (arXiv:2311.00571). arXiv. https://doi.org/10.48550/arXiv.2311.00571

Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., Schuhmann, C., Schmidt, L., & Jitsev, J. (2022). *Reproducible scaling laws for contrastive language-image learning* (arXiv:2212.07143). arXiv. https://doi.org/10.48550/arXiv.2212.07143

Chhikara, P., Chaurasia, D., Jiang, Y., Masur, O., & Ilievski, F. (2023). *FIRE: Food Image to REcipe generation* (arXiv:2308.14391). arXiv. https://doi.org/10.48550/arXiv.2308.14391

Choe, J., Oh, S. J., Lee, S., Chun, S., Akata, Z., & Shim, H. (2020). *Evaluating Weakly Supervised Object Localization Methods Right* (arXiv:2001.07437). arXiv.

https://doi.org/10.48550/arXiv.2001.07437

Dalmia, A., & Sia, S. (2021). *Clustering with UMAP: Why and How Connectivity Matters* (arXiv:2108.05525). arXiv. http://arxiv.org/abs/2108.05525

Devore, J. L., Berk, K. N., & Carlton, M. A. (2021). *Modern mathematical statistics with applications* (Third edition). Springer.

Dong, X., Li, H., Wang, X., Wang, W., & Du, J. (2023). CANet: Cross attention network for food image segmentation. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-023-17916-z

Ege, T., Shimoda, W., & Yanai, K. (2019). A New Large-scale Food Image Segmentation Dataset and Its Application to Food Calorie Estimation Based on Grains of Rice. *Proceedings of the 5th International Workshop on Multimedia Assisted Dietary Management*, 82–87. https://doi.org/10.1145/3347448.3357162

Erdem, K. (2022, July 21). *T-SNE clearly explained*. Medium. https://towardsdatascience.com/t-sne-clearly-explained-d84c537f53a

Ester, M., Kriegel, H., Sander, J., & Xu, X. (1996, August 2). *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*. Knowledge Discovery and Data Mining. https://www.semanticscholar.org/paper/A-Density-Based-Algorithm-for-Discovering-Clusters-Ester-Kriegel/5c8fe9a0412a078e30eb7e5eeb0068655b673e86

Gao, J., Tan, W., Ma, L., Wang, Y., & Tang, W. (2019). *MUSEFood: Multi-sensor-based Food Volume Estimation on Smartphones* (arXiv:1903.07437). arXiv. https://doi.org/10.48550/arXiv.1903.07437

Girdhar, R., El-Nouby, A., Liu, Z., Singh, M., Alwala, K. V., Joulin, A., & Misra, I. (2023). *ImageBind: One Embedding Space To Bind Them All* (arXiv:2305.05665). arXiv.

https://doi.org/10.48550/arXiv.2305.05665

Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge Distillation: A Survey. *International Journal of Computer Vision*, *129*(6), 1789–1819. https://doi.org/10.1007/s11263-021-01453-z

Guthrie, W. F. (2020). *NIST/SEMATECH e-Handbook of Statistical Methods (NIST Handbook 151)* [dataset]. [object Object]. https://doi.org/10.18434/M32189

Hao, S., Zhou, Y., & Guo, Y. (2020). A Brief Survey on Semantic Segmentation with Deep Learning. *Neurocomputing*, *406*, 302–321. https://doi.org/10.1016/j.neucom.2019.11.118

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *Null*. https://doi.org/10.1109/cvpr.2016.90

Helsedirektoratets kostråd. (2019, December 17). https://www.helsenorge.no/kosthold-og-ernaring/kostrad/helsedirektoratets-kostrad/

How UMAP Works—Umap 0.5 documentation. (n.d.). Retrieved May 15, 2024, from https://umap-learn.readthedocs.io/en/latest/how_umap_works.html

Huang, J., Jiang, K., Zhang, J., Qiu, H., Lu, L., Lu, S., & Xing, E. (2024). *Learning to Prompt Segment Anything Models* (arXiv:2401.04651). arXiv. https://doi.org/10.48550/arXiv.2401.04651

Ilharco, G., Wortsman, M., Wightman, R., Gordon, C., Carlini, N., Taori, R., Dave, A., Shankar, V., Namkoong, H., Miller, J., Hajishirzi, H., Farhadi, A., & Schmidt, L. (2021). *OpenCLIP* (v0.1) [Jupyter Notebook]. https://doi.org/10.5281/zenodo.5143773

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning: With applications in R* (Second edition). Springer.

https://doi.org/10.1007/978-1-0716-1418-1

Jiang, P.-T., Zhang, C.-B., Hou, Q., Cheng, M.-M., & Wei, Y. (2021). LayerCAM: Exploring Hierarchical Class Activation Maps for Localization. *IEEE Transactions on Image Processing: A Publication of the IEEE Signal Processing Society*, *30*, 5875–5888. https://doi.org/10.1109/TIP.2021.3089943

Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: A review and recent developments. *Philosophical Transactions Series A, Mathematical, Physical, and Engineering Sciences*, *374*(2065), 20150202. https://doi.org/10.1098/rsta.2015.0202

Jonathan Long, Long, J., Shelhamer, E., & Darrell, T. (2015). *Fully convolutional networks for semantic segmentation*. 3431–3440. https://doi.org/10.1109/cvpr.2015.7298965

K. Krishnaiah & P. Shahabudeen. (2012). *Applied Design of Experiment and Taguchi Metods*. PHI Learning Pvt. Ltd. https://www.phindia.com/Books/BookDetail/9788120345270/applied-design-of-experiments-and-taguchi-methods-shahabudeen

Khan, R., Kumar, S., Dhingra, N., & Bhati, N. (2021). The Use of Different Image Recognition Techniques in Food Safety: A Study. *Journal of Food Quality*, *2021*, 1–10. https://doi.org/10.1155/2021/7223164

Kiela, D., Bartolo, M., Nie, Y., Kaushik, D., Geiger, A., Wu, Z., Vidgen, B., Prasad, G., Singh, A., Ringshia, P., Ma, Z., Thrush, T., Riedel, S., Waseem, Z., Stenetorp, P., Jia, R., Bansal, M., Potts, C., & Williams, A. (2021). *Dynabench: Rethinking Benchmarking in NLP* (arXiv:2104.14337). arXiv. https://doi.org/10.48550/arXiv.2104.14337

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., & Girshick, R. (2023). *Segment Anything* (arXiv:2304.02643). arXiv. https://doi.org/10.48550/arXiv.2304.02643

Konstantakopoulos, F. S., Georga, E. I., & Fotiadis, D. I. (2024). A Review of Image-Based Food Recognition and Volume Estimation Artificial Intelligence Systems. *IEEE Reviews in Biomedical Engineering*, *17*, 136–152. https://doi.org/10.1109/RBME.2023.3283149

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, *25*. https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

L16b Orthogonal Array. (n.d.). Retrieved May 13, 2024, from https://www.york.ac.uk/depts/maths/tables/l16b.htm

Lan, X., Lyu, J., Jiang, H., Dong, K., Niu, Z., Zhang, Y., & Xue, J. (2024). FoodSAM: Any Food Segmentation. *IEEE Transactions on Multimedia*, 1–14. https://doi.org/10.1109/TMM.2023.3330047

Li, L. H., Zhang, P., Zhang, H., Yang, J., Li, C., Zhong, Y., Wang, L., Yuan, L., Zhang, L., Hwang, J.-N., Chang, K.-W., & Gao, J. (2022). *Grounded Language-Image Pre-Training*. 10965–10975. https://openaccess.thecvf.com/content/CVPR2022/html/Li_Grounded_Language-Image_Pre-Training_CVPR_2022_paper.html?ref=blog.roboflow.com

Li, S., Cao, J., Ye, P., Ding, Y., Tu, C., & Chen, T. (2024). *ClipSAM: CLIP and SAM Collaboration for Zero-Shot Anomaly Segmentation* (arXiv:2401.12665). arXiv. https://doi.org/10.48550/arXiv.2401.12665

Li, Y., Wang, H., Duan, Y., Xu, H., & Li, X. (2022). *Exploring Visual Interpretability for Contrastive Language-Image Pre-training* (arXiv:2209.07046). arXiv. https://doi.org/10.48550/arXiv.2209.07046

Lin, Y., Chen, M., Wang, W., Wu, B., Li, K., Lin, B., Liu, H., & He, X. (2023). *CLIP is Also an Efficient Segmenter: A Text-Driven Approach for Weakly Supervised Semantic Segmentation* (arXiv:2212.09506). arXiv. https://doi.org/10.48550/arXiv.2212.09506

Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Li, C., Yang, J., Su, H., Zhu, J., & Zhang, L. (2023). *Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection* (arXiv:2303.05499). arXiv. https://doi.org/10.48550/arXiv.2303.05499

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022). *A ConvNet for the 2020s* (arXiv:2201.03545). arXiv. https://doi.org/10.48550/arXiv.2201.03545

Lo, F. P. W., Sun, Y., Qiu, J., & Lo, B. (2020). Image-Based Food Classification and Volume Estimation for Dietary Assessment: A Review. *IEEE Journal of Biomedical and Health Informatics*, *24*(7), 1926–1939. https://doi.org/10.1109/JBHI.2020.2987943

Maaten, L. V. D., & Hinton, G. (2008). *Visualizing Data Using T-Sne*. *9*, 2579–2605.

Marin, J., Biswas, A., Ofli, F., Hynes, N., Salvador, A., Aytar, Y., Weber, I., & Torralba, A. (2019). *Recipe1M+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images* (arXiv:1810.06553). arXiv. https://doi.org/10.48550/arXiv.1810.06553

Matsuda, Y., Hoashi, H., & Yanai, K. (2012). Recognition of Multiple-Food Images by Detecting Candidate Regions. *Proceedings of the 2012 IEEE International Conference on Multimedia and Expo*, 25–30. https://doi.org/10.1109/ICME.2012.157

McInnes, L., Healy, J., & Melville, J. (2020). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction* (arXiv:1802.03426). arXiv. https://doi.org/10.48550/arXiv.1802.03426

Meenu, M., Kurade, C., Neelapu, B. C., Kalra, S., Ramaswamy, H. S., & Yu, Y. (2021). A concise review on food quality assessment using digital image processing. *Trends in Food*

Science & Technology, *118*, 106–124. https://doi.org/10.1016/j.tifs.2021.09.014

Meyers, A., Johnston, N., Rathod, V., Korattikara, A., Gorban, A., Silberman, N., Guadarrama, S., Papandreou, G., Huang, J., & Murphy, K. P. (2015). *Im2Calories: Towards an Automated Mobile Vision Food Diary*. 1233–1241. https://openaccess.thecvf.com/content_iccv_2015/html/Meyers_Im2Calories_Towards_an_ICCV_2015_paper.html

Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., & Mian, A. (2024). *A Comprehensive Overview of Large Language Models* (arXiv:2307.06435). arXiv. https://doi.org/10.48550/arXiv.2307.06435

Nguyen, H.-T., Cao, Y., Ngo, C.-W., & Chan, W.-K. (2024). FoodMask: Real-time food instance counting, segmentation and recognition. *Pattern Recognition*, *146*, 110017. https://doi.org/10.1016/j.patcog.2023.110017

Okamoto, K., & Yanai, K. (2021). UEC-FoodPix Complete: A Large-Scale Food Image Segmentation Dataset. In A. Del Bimbo, R. Cucchiara, S. Sclaroff, G. M. Farinella, T. Mei, M. Bertini, H. J. Escalante, & R. Vezzani (Eds.), *Pattern Recognition. ICPR International Workshops and Challenges* (pp. 647–659). Springer International Publishing. https://doi.org/10.1007/978-3-030-68821-9_51

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.-Y., Li, S.-W., Misra, I., Rabbat, M., Sharma, V., … Bojanowski, P. (2024). *DINOv2: Learning Robust Visual Features without Supervision* (arXiv:2304.07193). arXiv. https://doi.org/10.48550/arXiv.2304.07193

Parcalabescu, L., Trost, N., & Frank, A. (2021). *What is Multimodality?* (arXiv:2103.06304). arXiv. https://doi.org/10.48550/arXiv.2103.06304

Peter J. Woolf. (2009). *Chemical Process Dynamics and Controls*. University of Michigan Engineering Controls Group; Open Textbook Library. https://open.umn.edu/opentextbooks/textbooks/614

Petryk, S., Dunlap, L., Nasseri, K., Gonzalez, J., Darrell, T., & Rohrbach, A. (2022). *On Guiding Visual Attention With Language Specification*. 18092–18102. https://doi.org/10.1109/CVPR52688.2022.01756

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). *Learning Transferable Visual Models From Natural Language Supervision* (arXiv:2103.00020). arXiv. https://doi.org/10.48550/arXiv.2103.00020

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., & Sutskever, I. (2021). *Zero-Shot Text-to-Image Generation* (arXiv:2102.12092). arXiv. https://doi.org/10.48550/arXiv.2102.12092

Redmon, J., & Farhadi, A. (2017). *YOLO9000: Better, Faster, Stronger*. 7263–7271. https://openaccess.thecvf.com/content_cvpr_2017/html/Redmon_YOLO9000_Better_Faster_CVPR_2017_paper.html

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). *High-Resolution Image Synthesis with Latent Diffusion Models* (arXiv:2112.10752). arXiv. https://doi.org/10.48550/arXiv.2112.10752

Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

Rother, C., Kolmogorov, V., & Blake, A. (2004). "GrabCut": Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, *23*(3), 309–314. https://doi.org/10.1145/1015706.1015720

Salvador, A., Hynes, N., Aytar, Y., Marin, J., Ofli, F., Weber, I., & Torralba, A. (2017). *Learning Cross-Modal Embeddings for Cooking Recipes and Food Images*. 3020–3028. https://openaccess.thecvf.com/content_cvpr_2017/html/Salvador_Learning_Cross-Modal_Embeddings_CVPR_2017_paper.html

Schaller, R. R. (1997). Moore's law: Past, present, and future. *IEEE Spectrum*, *34*(6), 52–59. https://doi.org/10.1109/6.591665

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2020). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *International Journal of Computer Vision*, *128*(2), 336–359. https://doi.org/10.1007/s11263-019-01228-7

Sharma, U., Artacho, B., & Savakis, A. (2021). GourmetNet: Food Segmentation Using Multi-Scale Waterfall Features with Spatial and Channel Attention. *Sensors*, *21*(22), Article 22. https://doi.org/10.3390/s21227504

Simonyan, K., & Zisserman, A. (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition* (arXiv:1409.1556). arXiv. https://doi.org/10.48550/arXiv.1409.1556

Sinha, G., Parmar, K., Azimi, H., Tai, A., Chen, Y., Wong, A., & Xi, P. (2023). *Transferring Knowledge for Food Image Segmentation using Transformers and Convolutions* (arXiv:2306.09203). arXiv. https://doi.org/10.48550/arXiv.2306.09203

Smolensky, P. (1987). Connectionist AI, symbolic AI, and the brain. *Artificial Intelligence Review*, *1*(2), 95–109. https://doi.org/10.1007/BF00130011

Subramanian, S., Merrill, W., Darrell, T., Gardner, M., Singh, S., & Rohrbach, A. (2022). *ReCLIP: A Strong Zero-Shot Baseline for Referring Expression Comprehension* (arXiv:2204.05991). arXiv. https://doi.org/10.48550/arXiv.2204.05991

Thorwirth, Z. (2021, September 1). *AI Winter: The Highs and Lows of Artificial Intelligence*. History of Data Science. https://www.historyofdatascience.com/ai-winter-the-highs-and-lows-of-artificial-intelligence/

Tian, Y., Krishnan, D., & Isola, P. (2022). *Contrastive Representation Distillation* (arXiv:1910.10699). arXiv. https://doi.org/10.48550/arXiv.1910.10699

UECFoodPix,UECFoodPixComplete. (n.d.). Retrieved May 6, 2024, from https://mm.cs.uec.ac.jp/uecfoodpix/

Understanding UMAP. (n.d.). Retrieved May 15, 2024, from https://pair-code.github.io/understanding-umap/

Wang, H., Vasu, P. K. A., Faghri, F., Vemulapalli, R., Farajtabar, M., Mehta, S., Rastegari, M., Tuzel, O., & Pouransari, H. (2023). *SAM-CLIP: Merging Vision Foundation Models towards Semantic and Spatial Understanding* (arXiv:2310.15308). arXiv. https://doi.org/10.48550/arXiv.2310.15308

Wang, H., Wang, Z., Du, M., Yang, F., Zhang, Z., Ding, S., Mardziel, P., & Hu, X. (2020). *Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks*. 24–25. https://openaccess.thecvf.com/content_CVPRW_2020/html/w1/Wang_Score-CAM_Score-Weighted_Visual_Explanations_for_Convolutional_Neural_Networks_CVPRW_2020_paper.html

Wang, W., Min, W., Li, T., Dong, X., Li, H., & Jiang, S. (2022). A review on vision-based analysis for automatic dietary assessment. *Trends in Food Science & Technology*, *122*, 223–237. https://doi.org/10.1016/j.tifs.2022.02.017

Wang, X., Chen, G., Qian, G., Gao, P., Wei, X.-Y., Wang, Y., Tian, Y., & Gao, W. (2023).

Large-scale Multi-modal Pre-trained Models: A Comprehensive Survey. *Machine Intelligence Research*, *20*(4), 447–482. https://doi.org/10.1007/s11633-022-1410-8

Wang, Y., Chen, J., Ngo, C.-W., Chua, T.-S., Zuo, W., & Ming, Z. (2019). Mixed Dish Recognition through Multi-Label Learning. *Proceedings of the 11th Workshop on Multimedia for Cooking and Eating Activities*, 1–8. https://doi.org/10.1145/3326458.3326929

Wu, X., Fu, X., Liu, Y., Lim, E.-P., Hoi, S. C. H., & Sun, Q. (n.d.). *A Large-Scale Benchmark for Food Image Segmentation*. Retrieved May 6, 2024, from https://xiongweiwu.github.io/foodseg103.html

Wu, X., Fu, X., Liu, Y., Lim, E.-P., Hoi, S. C. H., & Sun, Q. (2021). A Large-Scale Benchmark for Food Image Segmentation. *Proceedings of the 29th ACM International Conference on Multimedia*, 506–515. https://doi.org/10.1145/3474085.3475201

Yin, Y., Qi, H., Zhu, B., Chen, J., Jiang, Y.-G., & Ngo, C.-W. (2023). *FoodLMM: A Versatile Food Assistant using Large Multi-modal Model* (arXiv:2312.14991). arXiv. https://doi.org/10.48550/arXiv.2312.14991

Zhai, X., Mustafa, B., Kolesnikov, A., & Beyer, L. (2023). *Sigmoid Loss for Language Image Pre-Training*. 11975–11986. https://openaccess.thecvf.com/content/ICCV2023/html/Zhai_Sigmoid_Loss_for_Language_Image_Pre-Training_ICCV_2023_paper.html

Zhai, X., Wang, X., Mustafa, B., Steiner, A., Keysers, D., Kolesnikov, A., & Beyer, L. (2022). *LiT: Zero-Shot Transfer with Locked-image text Tuning* (arXiv:2111.07991). arXiv. https://doi.org/10.48550/arXiv.2111.07991

Zhang, D., Han, J., Cheng, G., & Yang, M.-H. (2022). Weakly Supervised Object Localization and Detection: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(9), 5866–5885. https://doi.org/10.1109/TPAMI.2021.3074313

Zhang, M., Zhou, Y., Zhao, J., Man, Y., Liu, B., & Yao, R. (2020). A survey of semi- and weakly supervised semantic segmentation of images. *Artificial Intelligence Review*, *53*(6), 4259–4288. https://doi.org/10.1007/s10462-019-09792-7

Zhang, Q., Rao, L., & Yang, Y. (2021). *Group-CAM: Group Score-Weighted Visual Explanations for Deep Convolutional Networks* (arXiv:2103.13859). arXiv. https://doi.org/10.48550/arXiv.2103.13859

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2015). *Learning Deep Features for Discriminative Localization* (arXiv:1512.04150). arXiv. https://doi.org/10.48550/arXiv.1512.04150

9 - Appendices

9.1 - Appendix A - Code

The project code is available in a zipped archive file provided alongside the thesis document. Additionally, the code repository will be made available at this URL: https://github.com/Afterstone/ACIT5900-segmentation

9.2 - Appendix B - Food Image Dataset Taxonomy

Table 2

A Taxonomy of Food Image Datasets.

Dataset	Year	#Images	Classes	#Dishes	#Ingredients	Type	Source
American recipe dataset	2017	2,848	21			Recipe	Wang et al. 2022
Canteen dataset	2021	16,000	80			N/A	Wang et al. 2022
ChinaFood-100	2021	10,047	100			N/A	Wang et al. 2022
ChinaMartFood-109	2022	10,921	18			N/A	Wang et al. 2022
ChineseFood Net	2017	180,000	208			Classification	Konstantakopoulos et al. 2024
Diabetes	2014	5,420	11			Classification	Konstantakopoulos et al. 2024
ETH Food-101	2014	101,000		101		Classification	Wu et al. 2021
FfoCat	2019	58,962	156			Classification	Konstantakopoulos et al. 2024
FLD-469	2018	209,700	469			Classification	Konstantakopoulos et al. 2024
Food-101	2014	101,000	101			Classification	Konstantakopoulos et al. 2024
Food-11	2016	16,643	11			Classification	Konstantakopoulos et al. 2024
Food-975	2016	37,785	975			Classification	Konstantakopoulos et al. 2024
Food100	2012	14,361		100		Classification	Wu et al. 2021
Food201-Segmented	2015	12,625	201			Segmentation	Konstantakopoulos et al. 2024
Food254DB	2017	247,636	524			Classification	Konstantakopoulos et al. 2024

Food50	2010	5,000		50		Classification	Wu et al. 2021
Food50Seg	2020	5,000	50			Segmentation	Konstantakopoulos et al. 2024
Food85	2010	5,500		85		Classification	Wu et al. 2021
FoodAI-756	2019	400,000		756		Classification	Wu et al. 2021
FooDD	2015	3,000	30			Classification	Wang et al. 2022
FoodSeg103	2021	7,118		730	103	Segmentation	Wu et al. 2021
FoodSeg154	2021	9,490		730	154	Segmentation	Wu et al. 2021
FoodX-251	2019	158,846		251		Classification	Wu et al. 2021
Fruit-360	2018	90,483	131			Classification	Konstantakopoulos et al. 2024
FruitVeg-81	2017	15,737	81			Classification	Konstantakopoulos et al. 2024
Geo-Dish	2015	117,504		701		Classification	Wu et al. 2021
Indian Food Database	2017	5,000	50			Classification	Konstantakopoulos et al. 2024
Inselspital dataset	2015	1,620	248			Classification	Wang et al. 2022
ISIA Food-200	2019	197,323		200		Classification	Wu et al. 2021
Japanese recipe dataset	2017	4,877	15			Recipe	Wang et al. 2022
MAFood-121	2019	21,175	121			Classification	Konstantakopoulos et al. 2024
MedGRFood	2021	51,840	160			Classification	Konstantakopoulos et al. 2024
Mixed Dishes	2019	12,105	218			Segmentation	Konstantakopoulos et al. 2024
NIAD	2021	1,281	521			Recipe	Wang et al. 2022
Nutrinet	2017	225,953	520			Classification	Konstantakopoulos et al. 2024
PFID	2009	4,545		101		Classification	Wu et al. 2021
pic2kcal	2021	308,000	70,000			Recipe	Wang et al. 2022
Recipe1M	2017	1,000,000			1,488	Recipe	Wu et al. 2021
Recipe1M+	2019	14,000,000			1,488	Recipe	Wu et al. 2021
SUEC Food	2019	31,395	256			Segmentation	Konstantakopoulos et al. 2024
Sushi-50	2019	3,963		50		Classification	Wu et al. 2021
UEC Food256	2014	25,088		256		Classification	Wu et al. 2021

UEC-Food100	2012	9,132	100			Classification	Konstantakopoulos et al. 2024
UECFoodPix	2019	10,000		102		Segmentation	Wu et al. 2021
UECFoodPixComp.	2020	10,000		102		Segmentation	Wu et al. 2021
UNICT-FD1200	2016	4,754	1,200			Classification	Konstantakopoulos et al. 2024
UNICT-FG889	2014	3,583	889			Classification	Konstantakopoulos et al. 2024
UNIMIB 2016	2016	1,027	73			Classification	Konstantakopoulos et al. 2024
UPMC Food-101	2015	90,840		101		Classification	Wu et al. 2021
VegFru	2017	160,731	292			Classification	Konstantakopoulos et al. 2024
VIPER-FoodNet dataset	2021	14,991	82			Classification	Wang et al. 2022
Vireo Food-172	2016	110,241	172			Classification	Konstantakopoulos et al. 2024

Note. A taxonomy of 52 food image datasets. The taxonomy gathers dataset information from three survey papers and attempts to categorize them by year, type (Classification, segmentation and recipe), number of images and number of classes (Dishes and ingredients). The dataset type might not easily transfer to the other classes, since a segmentation dataset could have just food presence as a binary label. When either the number of dishes or ingredients are available, this is more specific than the number of classes, in which case the number of classes might be omitted. When the dataset type is not available, this means it does not map to the predefined classes, or the authors use ambiguous classes.