PICOOK: PHOTO INGREDIENTS TO COOKED DISH - GENERATING DISHES AND RECIPES FROM LEFTOVER INGREDIENTS

Jan Hagnberger, Andreas Glinka, Nikhil Bhavikatti
Department of Artificial Intelligence
University of Stuttgart
Universitätsstraße 32, 70569 Stuttgart, Germany
{st168021,st167563,st188468}@stud.uni-stuttgart.de

ABSTRACT

Food waste in the European Union is mainly caused by unused ingredients resulting from poor meal planning or ineffective grocery shopping habits. Improved food planning could help to reduce food waste, saving money and resources. However, effective food planning is time-consuming and requires creativity or a search for recipes. To address this, we propose a deep learning model named PICook, which takes photos of ingredients as input and generates images of potential dishes made from those ingredients. Additionally, the model outputs the name of a dish that could be made from those ingredients and a textual recipe for the dish. We assume that our model simplifies meal planning and helps to reduce food waste. The model development involves creating a pipeline based on foundation models like CLIP to generate a suitable dataset to train the image generation model (diffusion model) and adapting a text-to-image diffusion model for our purpose, followed by finetuning it on our dataset. We use a CLIP-based classifier and prompt a Large Language Model (LLM) to output dish names and recipes given a set of images of ingredients. We evaluate the components of our data generation pipeline and the diffusion model, demonstrating that our pipeline produces an appropriate dataset for our problem and that the diffusion model successfully generates images of things that look like food. Furthermore, we demonstrate that LLMs are able to generate dishes and recipes from ingredients. The code of the project is available at https://github.com/nikhilbhavikatti/PICooK.

1 INTRODUCTION

According to a 2022 Eurostat study (Eurostat, 2024), households are primarily responsible for food waste in the European Union, accounting for 55 % of the total food waste. In contrast, only 8 % of food waste was generated during primary production processes such as crop harvesting. The leading cause of household food waste is spoiled food or unused ingredients, often the result of suboptimal grocery shopping habits (Schmidt et al., 2019), like buying a cup of cream for a recipe, using only a fraction, and then forgetting about the rest. The authors suggest that more planned and need-based grocery shopping could mitigate this issue. However, planning meals in advance and smartly combining ingredients can be time-consuming and requires either creativity or a search for recipes. To address this problem, we propose a deep learning model called PICook. PICook, short for **P**hoto **I**ngredients to **Cook**ed Dish, takes photos of available ingredients, such as bananas and milk, as input and generates an image of a potential dish such as a milkshake that could be made with those ingredients. Additionally, our model outputs a dish name and recipe, given the images of ingredients. Our approach not only saves time but also eliminates the need for creativity or extensive search, making meal planning efficient and waste-free. Users can simply take photos of their ingredients and get an image of a dish, a dish name, and a recipe as output.

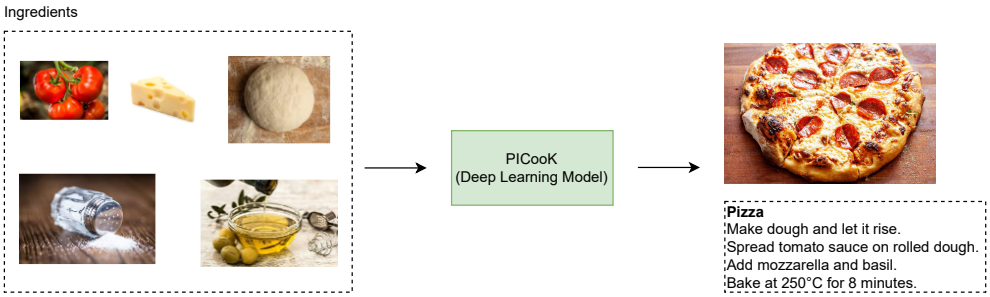

Figure 1: Our model takes as input a set of images of ingredients and generates an image of a dish as well as a textual description of the dish and a recipe.

2 Problem Statement

The first part of this project focuses on developing a deep learning model, f_θ, that processes a set of ingredient images $X = \{X_i\}_{i=1}^N$, where each $X_i \in \mathbb{R}^{w \times h \times 3}$ represents an RGB image. The model takes these images as input and generates an image of the prepared dish. Mathematically, it learns a function $f_\theta(X) = Y$, where Y corresponds to a synthesized image of the final dish. Here, θ represents the learnable parameters optimized during training. For example, given images of tomatoes and cheese, the model can generate an image of a pizza. Training this model requires a dataset $\mathcal{D} = \{(X, Y)_i\}$ consisting of paired inputs and outputs. The second part involves developing a separate model, $g_{\theta'}$, which takes ingredient images as input and outputs a dish name and its corresponding recipe. Formally, this is expressed as $g_{\theta'}(X) = Z$, where Z represents a textual sequence containing the dish and recipe. Figure 1 illustrates the problem our model aims to address. To achieve better performance and modularity, we opt for two distinct models: one for image generation (f_θ) and another for recipe generation ($g_{\theta'}$), which can later be combined for an integrated solution.

3 Method

The method is three-fold. Due to a lack of suitable datasets for training the images-to-image diffusion model on ingredients-dish image pairs, we first develop a pipeline based on existing foundation models to create a dataset. After the data collection process, we develop and train the model f_θ, which is also based on an existing foundation model, to map images of ingredients to an image of a dish. To generate the recipes (i.e., the model $g_{\theta'}$), we will simply reuse the zero-shot classifier and LLM from the dataset generation pipeline. Together, the diffusion model and recipe generator solve the problem described in the previous section.

3.1 Dataset Generation for Training the Diffusion Model

Our diffusion model should do images-to-image prediction, where the input images show ingredients and the output image is a dish made from those ingredients. However, existing datasets (Bień et al., 2020) contain only a list of ingredients as well as the name of the dish without any images. The best-suited and available datasets (Marin et al., 2019; Goel, 2020; Gari, 2023) for our use case contain only a list of ingredients, the name of the dish, and an image of the dish. On the other hand, there are a few datasets (Ahmed, 2024) that contain images of ingredients and the names of the ingredients. One approach for generating a dataset would be merging the two datasets (i.e., a dataset with a list of ingredients, dish name, and image of the dish with a dataset that contains images of ingredients). However, the ingredient dataset contains only a few different ingredients, which does not fully cover all ingredients used in the dish dataset. Furthermore, the dishes and ingredients in these datasets are often not common. Therefore, we design a pipeline to gather our own dataset. Figure 2 shows the pipeline that consists of the following three components. In short, the web scraper is used to collect images of dishes and ingredients from the Internet. As a second step, a zero-shot classifier is used to ensure that the scraped images show indeed the dishes or ingredients we are looking for. Finally, we

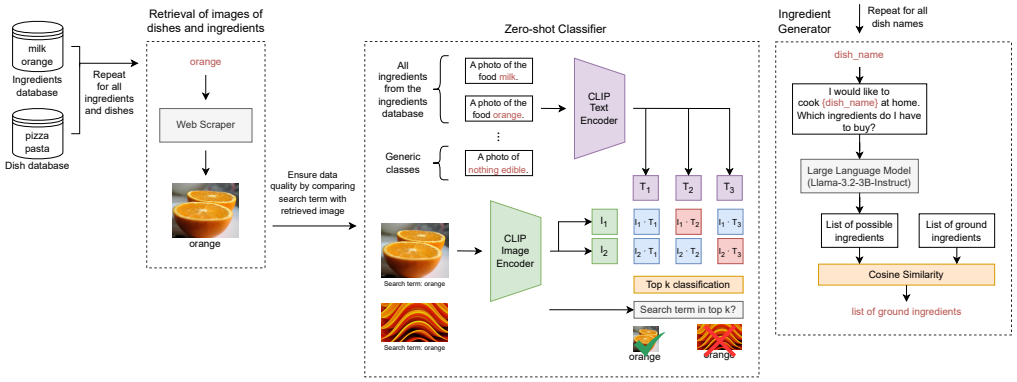

Figure 2: The data generation pipeline consists of three components. The image retrieval component searches the Internet for images of dishes and ingredients given a list with names of dishes and ingredients. The zero-shot classifier is used to ensure that the scraped images do not contain wrong or misleading images. An ingredient generator component generates a list of ingredients based on a given dish.

use a Large Language Model (LLM) to output the ingredients for a given dish and match them with the available ingredients. Combining the dish-ingredients mapping obtained from the LLM with the scraped images of dishes and ingredients yields our images-to-image dataset to train the diffusion model.

Image Retrieval. As a first step, we collect images of both dishes and ingredients to build a comprehensive dataset. To achieve this, we develop a custom web scraper using the Google Custom Search API for automated image retrieval. The scraper focuses on gathering high-quality images from publicly available sources by querying relevant keywords corresponding to various ingredients and dish names. For the ingredient dataset, we identify 292 unique ingredients, ranging from common items like tomatoes and milk to more specific ones like saffron and quinoa. Similarly, for dishes, we curate a list of 445 distinct dishes, including both everyday meals and international cuisines such as pasta, sushi, and biryani. The scraper aims to collect approximately 10 images per ingredient and dish, though the actual number varies depending on online availability. In total, the web scraper gathers over 2,594 ingredient images and 3,551 dish images from various sources. Several publicly available websites provided valuable sources of images, including Unsplash (Unsplash, 2013), Pexels (Pexels, 2015), Pixabay (Pixabay, 2010), Foodiesfeed (Foodiesfeed, 2014), Shopify Stock Photos (Shopify, 2011), and Wikimedia Commons (Foundation, 2004). These sites offered a rich variety of freely available images with proper licensing for research purposes.

Zero-Shot Classifier for Data Quality. Images from the Internet (e.g., from a Google image search) usually contain some wrong images. For instance, a Google image search for the search term/ keyword "orange" could yield some orange backgrounds or the fruit orange. To avoid having wrong images in our dataset, we utilize a CLIP-based zero-shot classifier to ensure that the scraped images do not contain wrong images. The zero-shot classifier classifies images into different classes of ingredients, dishes, and generic classes like "non-edible" or "person". By running the classifier on all scraped images, we first classify the image and check whether the class matches the search term (i.e., the expected class, the image should represent). If there is no match, the image is probably wrong or misleading and we can remove it. This essentially allows us to generate a dataset with images of ingredients and dishes that do not contain wrong images.

The classifier is based on CLIP (Radford et al., 2021) and is used for inference only to do zero-shot classification. The image encoder of CLIP is used to obtain a latent representation of the image. The text encoder is used to obtain latent representations of dishes and ingredients that are encoded as text like "a photo of the food orange". We also use some generic labels such as "a photo of a person", "a photo of something non-edible". By computing the cosine similarity between the image and text embeddings, we get a quantitative measure of how well the image matches the textual description.

We compute a `softmax` function on top of the cosine similarities and do a top-k classification. If the research term (expected label) is not contained in the top-k results of the classes, there is a high probability that the image is wrong and we mark the image as wrong or misleading and remove it. We set $k = 3$ since it empirically produces good results.

Ingredient and Dish Generator. After creating a dataset containing images of ingredients and dishes as well as their corresponding names, we move to developing the ingredient generator. The primary objective is to produce a mapping between dishes and their associated ingredients to later generate the dataset for training the diffusion model. To accomplish this, we implement an ingredient generator, which predicts ingredients based on dish names, and a dish generator that generates dishes from given ingredients.

For both tasks, we utilize Meta's Llama-3.2-3B-Instruct model (Touvron et al., 2023) as the main large language model (LLM). In both generators, the LLM is queried by first setting a context tailored to the specific problem, followed by providing a dish name or list of ingredients as input. The output is a list of ingredients or a single dish name, respectively. To match the generated ingredients with the list of possible ingredients (ground ingredients), we compute the cosine similarity between the outputted ingredients and our ground ingredients and pick the ground ingredient with the highest cosine similarity. For instance, the LLM could output "Basmati rice" and we only have the ground ingredient "rice" (i.e., we have only images of "rice"). Thus, we have to match them. The dish generator utilizes the same principle to match the generated dishes with our list of possible dishes.

We evaluated the performance of both generators to determine which performed best, with the results presented in section 5. In our experiments, we found that the output of the dish generator was partially misleading, containing ingredients in the dish names. Based on that and the evaluation, we selected the ingredient generator for the task, as it produced clearer and more reliable outputs.

Everything Together. Putting all three components together yields the data generation pipeline. The quality of the images of the ingredients and dishes (web scraper) is ensured by removing the wrong images using the zero-shot classifier. Then, the images of the dishes are combined with the images of ingredients by utilizing the ingredients-dish mapping from the ingredient generator. As a result, we get a dataset $\mathcal{D}$ with samples $(X, Y)_i$ where X is a set of images of ingredients and Y is an image of a cooked dish.

3.2 Image Generation Model

We utilize a generative model to generate images of cooked dishes based on the input images of ingredients. Possible generative architectures are Variational Autoencoders (VAEs) (Kingma & Welling, 2022), Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), and Diffusion Models (Ho et al., 2020). We decided to use a diffusion model since VAEs often produce low-quality images and GANs suffer from problems such as mode collapse and instabilities (Saxena & Cao, 2023). In particular, we employ a latent diffusion model because it is more efficient than traditional diffusion models. This efficiency arises from the fact that the diffusion and denoising processes happen within a smaller latent space rather than the larger original image space. To generate images of a dish based on images of ingredients, we utilize a conditional diffusion model that allows guiding the diffusion process with a conditioning factor z. To reduce the computational costs, we use a pretrained Stable Diffusion model (Rombach et al., 2022) (Stable Diffusion V1.4) for text-to-image predictions, modify the model architecture for our problem of images-to-image predictions, and finetune it on the generated dataset.

3.2.1 Conditional Latent Diffusion Model

Stable Diffusion V1.4, a conditional latent diffusion model, takes as input a text (i.e., sequence of words) and outputs an image matching the textual description. As a first step, the text is encoded into a latent representation $z \in \mathbb{R}^{44\times768}$ where 44 corresponds to the padded sequence length and 768 to the latent dimension of each token. The encoding is done with the text encoder of CLIP which is based on a Transformer (Vaswani et al., 2023). The representation z guides the diffusion process to generate an image that aligns with the textual description by conditioning the denoising UNet (Ronneberger et al., 2015) on z. In Stable Diffusion, cross-attention (Vaswani et al., 2023) serves as

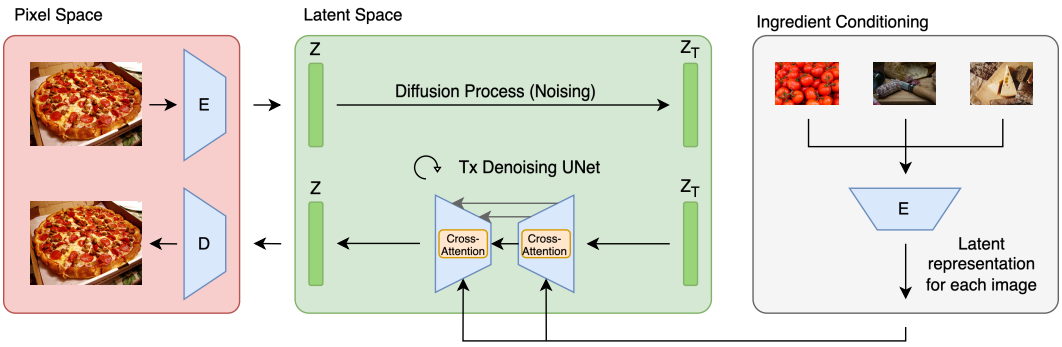

Figure 3: The image generation model is based on conditional latent diffusion. The conditioning mechanism is implemented by encoding each image using the image encoder of CLIP. The encoded images are used to obtain the key and value vectors for cross-attention. The query vector will be constructed by the denoising UNet and allows it to "query the ingredients" to guide the generation process.

the conditioning mechanism, meaning the key and value are derived from z, while the query is constructed by the denoising UNet. Intuitively, cross-attention allows the UNet to "query" information from the conditioning factor (i.e., the textual description). To use the Stable Diffusion model for our purpose, we modify it to do images-to-image prediction which includes adding an encoder that encodes input images into a latent representation as well as incorporating the latent representation into the denoising UNet for conditioning. Figure 3 shows the conditional latent diffusion model with the ingredient conditioning based on the image encoder of CLIP and cross-attention for conditioning.

Image Encoder. To condition the diffusion model on a set of images, we first apply an encoder to encode the input images into a latent space. We use the image encoder of CLIP with a pooling operator to obtain a latent representation $z_i \in \mathbb{R}^{768}$ for each image. Thus, the final latent representation is $z \in \mathbb{R}^{n \times 768}$ where n represents the number of images of ingredients. We opt for the CLIP image encoder because we hope that the *pooled image features*, produced by the CLIP image encoder and which we use for conditioning, have a high similarity to the corresponding *text features* generated by the CLIP text encoder and which have been used in the Stable Diffusion model training. For instance, the *pooled image features* of an image of a banana should have a high similarity to the *text features* of the sentence "a photo of a banana". It is important to note that this is not automatically given, even if CLIP has been trained with the contrastive learning objective for the CLIP text and image encoder. The CLIP encoders output features, pooled features, and pooled features with a linear projection. During the contrastive learning of CLIP, the (dis)similarity between a projection of the *pooled text features* and *pooled image features* was maximized/ minimized. Therefore, it is not given by construction that the *features* and *pooled features* have the same (dis)similarity as the pooled features with projection. We condition the UNet on *pooled features* while it has been trained on the *text features* and not on the *pooled text features* during the Stable Diffusion training. Using the *image features* instead of the *pooled image features* for conditioning has the same problem and increases the computational complexity because it is a sequence of vectors and not only a single vector per image.

Conditioning Mechanism. The latent representation z, which describes the ingredients, is fed into the conditioning mechanism of the denoising UNet. We keep the cross-attention conditioning mechanism to avoid drastic changes in the neural architecture. For instance, replacing cross-attention with a different conditioning mechanism such as Adaptive Layer Normalization (AdaLN)(Perez et al., 2017) could yield problems during the finetuning since the model was pretrained with cross-attention and not AdaLN. The denoising UNet is now conditioned on a sequence of images instead of a sequence of tokens (i.e., text) as in the original setup.

Finetuning of Pretrained Diffusion Model. We freeze the CLIP image encoder and train only the query, key, and value weight matrices of the cross-attention mechanism of the denoising UNet to

reduce computational and memory complexity. Furthermore, we assume that finetuning the weight matrices of the cross-attention is sufficient because we have mainly changed the conditioning factors of the model and kept the remaining parts untouched. The finetuning is done using a Low-Rank Adaptation (LoRa) (Hu et al., 2021).

Diffusion Model from Scratch. We also considered building and training a latent diffusion model based on an UNet (Ronneberger et al., 2015) from scratch. However, we didn't pursue it due to limited computing resources.

3.3 Recipe Generator

Up to this point, we only considered the first part of the project about the image generation, but not the recipe generation. Thus, we have to develop the recipe generator $g_{\theta'}$ that takes as input a set of images and outputs the dish name and the recipe as text. We simply use our CLIP-based zero-shot classifier to first classify the input images into ingredients (e.g., an image of olive oil will be classified as olive oil). After that, we prompt the dish generator (LLM) to output a dish name and recipe given the ingredients. The diffusion model and recipe generator together build the PICook model that suggests dishes and recipes as well as generates an image of a dish given images of ingredients.

4 Used Resources

To summarize, we use the following resources. The web scraper is a custom web scraper based on the Google Image Search API that scrapes images from webpages like Unsplash (Unsplash, 2013), Pexels (Pexels, 2015), and Pixabay (Pixabay, 2010). The zero-shot image classifier is based on the image-text foundation model CLIP (Radford et al., 2021) and the ingredient generator is based on the large language model Llama (Touvron et al., 2023). We customize a Stable Diffusion (Rombach et al., 2022) model for the ingredients-to-dish image generation.

5 Experiments

We design our experiments to answer the following research questions:

- (RQ1): How many random or bad images are in the scraped images?
- (RQ2): How good is the zero-shot classifier in detecting wrong images in the scraped dataset?
- (RQ3): How good are the ingredients and dishes generated by the dish/ingredient generator?
- (RQ4): How good is the data generation pipeline?
- (RQ5): How good is the diffusion model in generating photorealistic images of dishes given a set of ingredients?
- (RQ6): How does the performance of our custom diffusion model compare to a state-of-the-art pretrained model like Stable Diffusion for generating dish images?
- (RQ7): How do final results (generated image, dish name, and recipe) look?

5.1 RQ1: Quality of Scraped Images

We generate a list of 292 different ingredients and 445 different dishes and scrape corresponding images from the Internet. We try to scrape 10 images for each ingredient and dish. However, this is not always possible since sometimes fewer images are available, especially for some special dishes. Therefore, we got in total 2'594 images of ingredients and 3'551 images of dishes. Using the images without any further processing step to ensure data quality would yield a dataset with poor quality because some of the scraped images show something different. Figure 4 shows some wrong examples. Thus, we apply the CLIP-based zero-shot classifier to remove wrong images which do not match the keyword (e.g., the image of the orange structure from Figure 4 has a small similarity with

Figure 4: Selection of wrong images in the scraped images. For example, the right image shows some orange structure instead of the fruit orange.

the text "a photo of the food orange" and will be removed). After removing wrong and corrupted images, we got a dataset with 2'425 images of ingredients (168 wrong and 1 corrupted image have been removed) and 2'646 images of dishes (904 wrong and 1 corrupted image have been removed). We combine the cleaned images of the ingredients with the dishes using the dish-ingredient mapping obtained by the ingredient generator.

5.2 RQ2: Performance of the Zero-shot Ingredient Classifier

We evaluate the performance of our zero-shot image classifier to determine its effectiveness in filtering misleading or irrelevant images. For this evaluation, we utilize a combined dataset of 8000 images, including 4000 labeled ingredient images from the Recipe Ingredients Image Dataset (Ahmed, 2024) and 4000 randomly selected images from the ImageNet-mini dataset (Figotin, 2019). For our evaluation, a subset of this dataset was selected, and the images were assigned random labels corresponding to the 40 ingredient classes, simulating noisy and potentially misleading inputs.

The evaluation process focuses on the classifier's ability to accurately assign ingredient-based labels to the images. Using the CLIP-based zero-shot classifier, we process the images in batches, generate predictions based on a predefined set of labels. The classifier's performance is measured using standard metrics such as accuracy, precision, recall, and F1-score. While multiple top-k prediction scenarios were analyzed during testing, we report the results for the strictest case, top-k=1, where the classifier outputs a single most confident label for each image. In addition to this, we conducted a similar experiment using dish images instead of ingredient images. These dish images, sourced from the Food-101 dataset (Mäder, 2014), were categorized into 40 dish categories, allowing us to test the classifier's ability to generalize its predictions to a different type of input data. The results for the dish image experiment were comparable to those obtained with ingredients, further demonstrating the classifier's reliability. The results are summarized in Table 1.

The classifier's performance underscores its effectiveness in cleaning datasets by identifying and removing irrelevant or incorrect images. Its high recall ensures that misleading inputs are rarely overlooked, which is critical for maintaining dataset quality. While the slightly lower precision indicates some over-inclusion of irrelevant images, this is an acceptable compromise given the priority to avoid false negatives. The balanced performance reflected in the metrics highlights the suitability of the classifier for this application.

In conclusion, the zero-shot classifier is a robust tool for filtering images, making it well-suited for generating clean datasets for training. Future work will involve refining the classifier to improve precision without compromising recall. This could include optimizing the text prompts used in zero-shot classification or testing the model on larger and more diverse datasets to assess its generalization capabilities. Enhancing these aspects could further solidify its role as an indispensable tool in dataset preparation.

Dataset	**Accuracy**	**Precision**	**Recall**	**F1-score**
Ingredients	0.91	0.85	0.99	0.91
Dishes	0.93	0.87	0.99	0.93

Table 1: Comparison of Metrics for Zero-shot Ingredient Classifier

5.3 RQ3: Performance of the Ingredient and Dish Generator

We assess the performance of our dish and ingredient generator by conducting a test in which we generate multiple dishes and their corresponding ingredients, comparing them against the ground truth. For this evaluation, we utilize the recipes3k Gari (2023) dataset and generate a total of 1000 dishes and ingredients using our generators.

The primary goal is not to produce exact matches of dishes or ingredients but to capture their overall concept accurately. Therefore, to evaluate the effectiveness of the generators, we use the BLEU score and cosine similarity as performance metrics, calculating the average performance across all 1000 generated entries. The results are summarized in Table 2.

To compute the cosine similarity, we extract embeddings for both the ingredients and dishes using the sentence transformer all-MiniLM-L6-v2 Transformers (2020). Our findings reveal that the ingredient generator achieves a high overall cosine similarity, suggesting that the generated ingredients effectively reflect the intended ideas of the real ingredients. In contrast to the dish generator, the BLEU score of the ingredient generator is relatively low, primarily due to the lack of a specific order in the generated ingredients. When investigating the outputs of the dish generator, the dishes seem rather good. This is also reflected in the scores, however, due to the nature of the same ingredients having multiple possible dishes, sometimes a different dish is generated than the one given in the dataset. Although those generated dishes are still correct, meaning they consist mostly of the same ingredients, this became a bigger issue when searching for images of the generated dish names, some of which partially consisted of the ingredients themselves. An example entry for this problem is "Chestnut and Potato Gratin with Thyme and Walnuts" which should have been "Mushroom and Potato Soup". This lead to bad image samples of the dishes when scrapping and deteriorated the dataset with more ingredient images.

In summary, the performance of our generators is sufficient for the purpose of generating plausible dishes and ingredients. Rather than aiming to recreate exact dishes, our goal is to generate reasonable or similar alternatives based on the given ingredients. However, when searching for images of the dishes the ingredient generator performed better, resulting in clear images of ingredients and having an overall higher cosine similarity. Thus, we used it for our purpose of creating the dataset.

Generator	**Average Cosine Sim**	**Average BLEU**
Dish	0.57	0.12
Ingredient	0.78	0.02

Table 2: Comparison of Metrics for Dish and Ingredient Generator

An example output from the Ingredient Generator can be seen in Table 3. It shows that even with a cosine similarity of 0.78 our generated ingredients are mostly correct, differing only in small details like in butter and unsalted butter. Some ingredients are missing, however the overall idea is captured well with respect to the dish.

True Ingredients	Generated Ingredients
self-raising flour	all-purpose flour
golden caster sugar	granulated sugar
butter	unsalted butter
eggs	large eggs
bicarbonate of soda	baking powder
	salt
bar white chocolate	cocoa powder
bar dark chocolate	
dried sour cherry	cherry jam
	cherries
natural yogurt	

Table 3: True and generated ingredients for Choch-Cherry-Muffins with a cosine similarity of 0.78

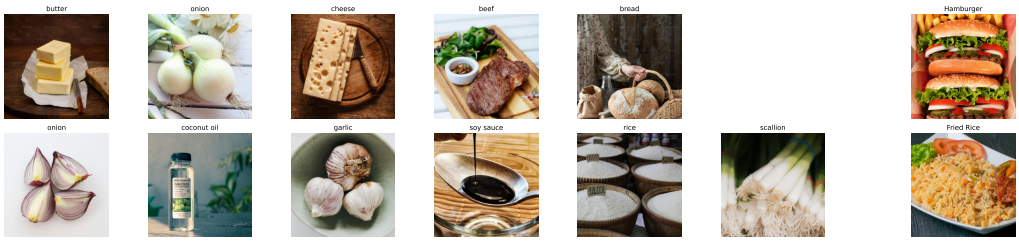

Figure 5: Each row shows one sample of our dataset. The left side shows the ingredients (model's input) and the right side the dish (desired target).

5.4 RQ4: Data Generation Pipeline

After assessing the performance of each component, we can now put the dataset generation pipeline together as described in Section 3. Figure 5 shows two random samples from our dataset. Each row represents one entry of our dataset to train the diffusion model. The left side shows the images of ingredients (input for the model) and the right side the dish (target). The zero-shot classifier ensures that the images match the needed tags of the ingredient or dish. The LLM generates a recipe or mapping for dishes to ingredients by which the images are put together.

5.5 RQ5: Evaluation of the Diffusion Model

We train the diffusion model on our dataset containing 2'208 samples of ingredient-dish image pairs. The training process comprises 112 epochs with a batch size of 32 which took approximately one day on an NVIDIA A100 GPU. We test the model on our test set of 552 samples with 30 denoising steps for the image generation. We split the evaluation into the following sections.

5.5.1 Can the model generate images of dishes and food?

First, we qualitatively evaluate whether the images show dishes and are good-looking or not. The results demonstrate that the diffusion model is indeed able to generate images of "things" that have a high similarity with food or dishes. For instance, Figure 6 shows some good examples that are generated by the diffusion model. On the one hand, the generated images show something that looks like a dish and on the other hand, the images are good-looking. However, the model does not always generate good-looking images of dishes and sometimes generates images of things that do not have anything in common with food or are bad-looking. Figure 7 shows some bad examples. These bad examples show some abstract concepts, people, or text that looks like a restaurant menu. Consequently, the model sometimes generates bad-looking images or images that have nothing in common with food or dishes. Figure 10 shows 36 randomly sampled images generated by the diffusion model.

5.5.2 How many images are good (food image) and bad (non-food image)?

Motivated by the qualitative evaluation from the previous section, we also investigate the ratio of good-looking dish images and bad examples that either show something different than a dish or a bad-looking dish. We evaluate the performance with a human assessment of 64 randomly generated images which yields that approximately 50 % of the generated images show some good-looking food-like things. In addition, we automatically evaluate the ratio using CLIP with zero-shot classification. The automatic evaluation shows that 51 % are good-looking food images which matches the results of the human assessment.

5.5.3 Does the model learn the relationship between ingredients and dish?

As a third criterion, we test if the model learns the relationship between the input images of ingredients and the generated images. Figure 11 shows the generated images with a fixed random noise while reducing the ingredients to condition on. It is visible that the conditioning affects the generated image (items and bowls disappear). However, the difference in the generated images is not as

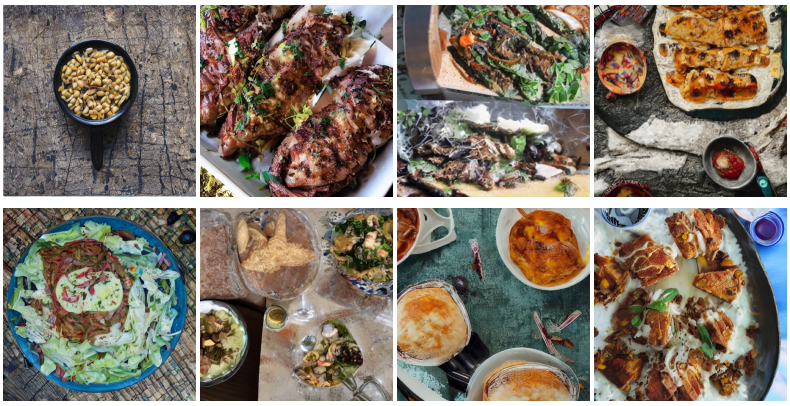

Figure 6: Good example images generated by the diffusion model. Each image shows something that looks like a dish. The model took 30 denoising steps to generate the images.

Figure 7: Bad example images generated by the diffusion model. Each image shows something totally different from a dish. The first image shows some abstract concept, the second two persons, the third a place outside, and the last one a text similar to a restaurant menu. The model took 30 denoising steps to generate the images.

a human would expect. Thus, we conclude that the model is not fully able to learn the relationship between the ingredients and the dish. We assume that either the quality of our dataset is not sufficient (it still contains some misleading ingredient-dish samples and it is hard to learn the relationship between the ingredients and the dish) or fine-tuning the weight matrices of the attention mechanism is not sufficient and more components of the UNet and CLIP image encoder should be trained.

5.5.4 Quantitative Results

We also compute the Frechet Inception Distance (FID) score on the test set which yields a value of 16.73 for our diffusion model. The FID score evaluates the quality of images and measures the similarity between the generated and ground truth images. The value range is $[0, \infty)$ and lower values for the FID score are better. In addition, we compute the CLIP score (Hessel et al., 2022) between the generated image and the textual description of the dish on the test set which yields a value of 24.34. The CLIP score compares the semantics of the generated image by comparing the image with the textual name of the dish name. The value range of the CLIP score is $[0, 100]$ and higher values are better. Table 4 compares our model against two Stable Diffusion text-to-image models that we either prompt on the dish name or a list of ingredient names. The table shows that the baseline models are much better than our model which is not surprising since the fine-tuning might have "destroyed" the pretrained model.

5.5.5 Qualitative Results

Figure 8 shows randomly selected samples with the ingredient images, which are used to condition the model on, as well as the generated image and the ground truth image. The generated images of the diffusion model seem relatable, except for the last sample (last low). The generated image of the third example (third row) is even better than the ground truth image from our dataset. However,

Figure 8: Each row represents one sample consisting of the input ingredient images (images on the left side), generated image (second-last image), and ground truth image (last image). The model took 30 denoising steps to generate the images.

learning the relationship between images of ingredients and a dish is hard because often ingredients are not directly visible in the dish and the mapping is ambiguous (i.e., the same ingredients can be combined to create many different dishes).

5.6 RQ6: Comparison with Stable Diffusion

In this experiment, we compare the results of our custom PICook diffusion model with the pretrained Stable Diffusion model (CompVis, 2021) for the task of generating dish images based on textual prompts. The objective was to evaluate the quality of the generated images in terms of their similarity to real dish images, using the Frechet Inception Distance (FID) score as the evaluation metric and to measure the semantics using the CLIP score.

We designed two types of experiments for generating dish images using Stable Diffusion. In the first experiment, the model was provided with prompts based on the ingredients of each dish, where a textual prompt of the form *"Generate an image of a dish with ingredient1, ingredient2, ..."*, listing the primary ingredients, was used to generate 170 images. In the second experiment, prompts based on the dish names were used. In this case, a prompt of the form *"Generate an image of the dish dish_name"* was provided to generate another set of 170 images.

Both sets of generated images were then compared against the same set of 170 real dish images using the FID score. The FID score obtained for the ingredient-based prompts was 2.23, while the FID score for the dish name-based prompts was 1.86, indicating a high level of similarity between the generated and real images. These low FID scores suggest that Stable Diffusion is capable of producing highly realistic dish images when given appropriate textual prompts.

While the low FID scores are promising, it is important to note that these scores alone do not guarantee that the generated images are always representative of the actual dishes, especially when prompts based on ingredients are used. Future work could focus on fine-tuning Stable Diffusion specifically on food-related datasets, which may further improve both the quality and relevance of the generated images.

Model	FID Score (↓)	CLIP Score (↑)
Stable Diffusion text-to-image conditioned on dish name	**1.86**	**25.29**
Stable Diffusion text-to-image conditioned on ingredient names	2.23	25.14
PICook Diffusion images-to-image (ours)	16.73	24.34

Table 4: Comparison of FID and CLIP scores for two Stable Diffusion text-to-image baseline models, which were prompted either with the dish name or a list of ingredient names, and our PICook images-to-image diffusion model directly takes images of ingredients as input. The FID score ranges from 0 to ∞ and lower values are better. The CLIP score ranges from 0 to 100 and higher values are better.

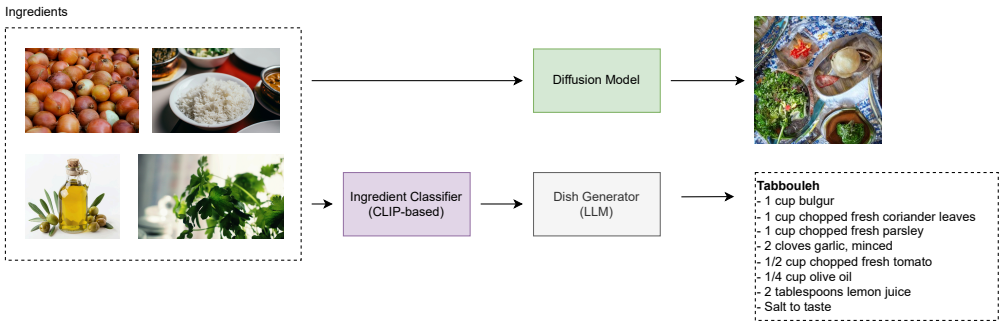

Figure 9: Final example prediction. The input is a set of images of ingredients and the diffusion model directly outputs an image of a dish. The dish name and recipe are generated by first classifying the ingredient images using a CLIP-based classifier and prompting an LLM to generate a dish and a recipe for the dish.

5.7 RQ7: Final Results

Our final proposed pipeline is presented in Figure 9. We use our diffusion model with direct image inputs to generate an image of a possible dish. At the same time, we reuse some of our components to generate a possible recipe. The ingredient images are first put into our zero-shot classifier to retrieve textual tags or ingredient names for the images. Afterwards, the list of ingredients is passed into a slightly changed version of our dish generator to generate a possible recipe. Therefore, we obtain a possible recipe for a dish and a corresponding image as an output.

6 Conclusion

In this research, we addressed the generation of possible dish images using available ingredient images by utilizing a latent diffusion model. To train the model, we collected a custom dataset by automatically scraping the Internet and ensuring data quality with CLIP-based filtering. An ingredient generator was employed to generate a list of base ingredients for specific dishes by querying a large language model. Our experiments demonstrate that the implemented approaches effectively supported the dataset generation process. For the training process, we used the CLIP image encoder to extract pooled latent features from the ingredient images, which were then used to condition the UNet of the diffusion model. This conditioning was implemented using cross-attention, and only the query, key, and value weight matrices of the cross-attention mechanism were fine-tuned to minimize computational complexity. The dataset, which mapped ingredient images to corresponding dish images, was used to train the diffusion model for 112 epochs with a batch size of 32. The trained model achieved a Frechet Inception Distance (FID) of 16.73 and a CLIP score of 24.34, demonstrating its ability to generate visually appealing dish images, with 51% evaluated as high-quality food-like outputs. We also built a recipe generator that outputs a dish name and recipe given a set of ingredient images. The model is based on our CLIP-based zero-shot classifier and the dish generator from the data generation pipeline. Thus, we demonstrated that LLMs can generate suitable recipes. Together, the models can generate an image of a dish and a recipe from images of ingredients. While our results seem promising, there is still room for improvement.

7 Future Work

Future work could include scraping a bigger dataset for the diffusion model and using a human quality check. Although we use our effective classifier to detect wrong images, the dataset still contains wrong or misleading samples. To improve the diffusion model, it could also be helpful to finetune more components instead of freezing them and or to use a small CNN to generate the conditioning vectors instead of using the CLIP image encoder. It could also be interesting to use a small mapping network between the CLIP embeddings and the conditioning input of the denoising UNet. Currently, the diffusion model and recipe generator are not connected, which could lead to

misleading outputs (i.e., the dish image is completely different from the recipe). Thus, it could be helpful to connect both models by also conditioning the diffusion model on the dish name.

REFERENCES

Fasih Ahmed. Recipe Ingredients Image Dataset — kaggle.com. https://www.kaggle.com/datasets/fasihcs/recipe-ingredients-image-dataset, 2024. [Accessed 25-10-2024].

Michał Bień, Michał Gilski, Martyna Maciejewska, Wojciech Taisner, Dawid Wisniewski, and Agnieszka Lawrynowicz. RecipeNLG: A cooking recipes dataset for semi-structured text generation. In *Proceedings of the 13th International Conference on Natural Language Generation*, pp. 22–28, Dublin, Ireland, December 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.inlg-1.4.

CompVis. Stable diffusion v1.4, 2021. URL https://huggingface.co/CompVis/stable-diffusion-v1-4. Accessed: 2025-01-02.

Eurostat. Déchets alimentaires et prévention des déchets alimentaires par activité de la nace rév. 2 - tonnes de masse fraîche, 2024. URL https://ec.europa.eu/eurostat/databrowser/product/page/ENV_WASFW.

Ilya Figotin. Imagenet-mini dataset. https://www.kaggle.com/datasets/ifigotin/imagenetmini-1000, 2019.

Foodiesfeed. Foodiesfeed - free food images, 2014. URL https://www.foodiesfeed.com/.

Wikimedia Foundation. Wikimedia commons - free media repository, 2004. URL https://commons.wikimedia.org/.

Crispen Gari. Food Recipes — kaggle.com. https://www.kaggle.com/datasets/crispen5gar/recipes3k, 2023. [Accessed 25-10-2024].

Sakshi Goel. Food Ingredients and Recipes Dataset with Images — kaggle.com. https://www.kaggle.com/datasets/pes12017000148/food-ingredients-and-recipe-dataset-with-images, 2020. [Accessed 25-10-2024].

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. URL https://arxiv.org/abs/1406.2661.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning, 2022. URL https://arxiv.org/abs/2104.08718.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *CoRR*, abs/2006.11239, 2020. URL https://arxiv.org/abs/2006.11239.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. URL https://arxiv.org/abs/1312.6114.

Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.

K. Mäder. Food-101 dataset. https://www.kaggle.com/datasets/kmader/food41, 2014.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer, 2017. URL https://arxiv.org/abs/1709.07871.

Pexels. Pexels - free stock photos, 2015. URL https://www.pexels.com/.

Pixabay. Pixabay - free images and videos, 2010. URL https://pixabay.com/.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. URL https://arxiv.org/abs/2103.00020.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

Divya Saxena and Jiannong Cao. Generative adversarial networks (gans survey): Challenges, solutions, and future directions, 2023. URL https://arxiv.org/abs/2005.00065.

Thomas Schmidt, Felicitas Schneider, Erika Claupein, Thomas Schmidt, Felicitas Schneider, and Erika Claupein. Food waste in private households in germany – analysis of findings of a representative survey conducted by gfk se in 2016/2017 –. 2019. doi: 10.22004/AG.ECON.290529. URL https://ageconsearch.umn.edu/record/290529.

Shopify. Shopify stock photos, 2011. URL https://www.shopify.com/stock-photos.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Sentence Transformers. all-minilm-l6-v2. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2, 2020. Accessed: 2024-12-04.

Unsplash. Unsplash - beautiful free images pictures, 2013. URL https://unsplash.com/.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

A Appendix

Figure 10 shows 36 randomly selected images generated by the diffusion model using 30 denoising steps. Figure 11 shows how the output is affected when ingredients are removed or added.

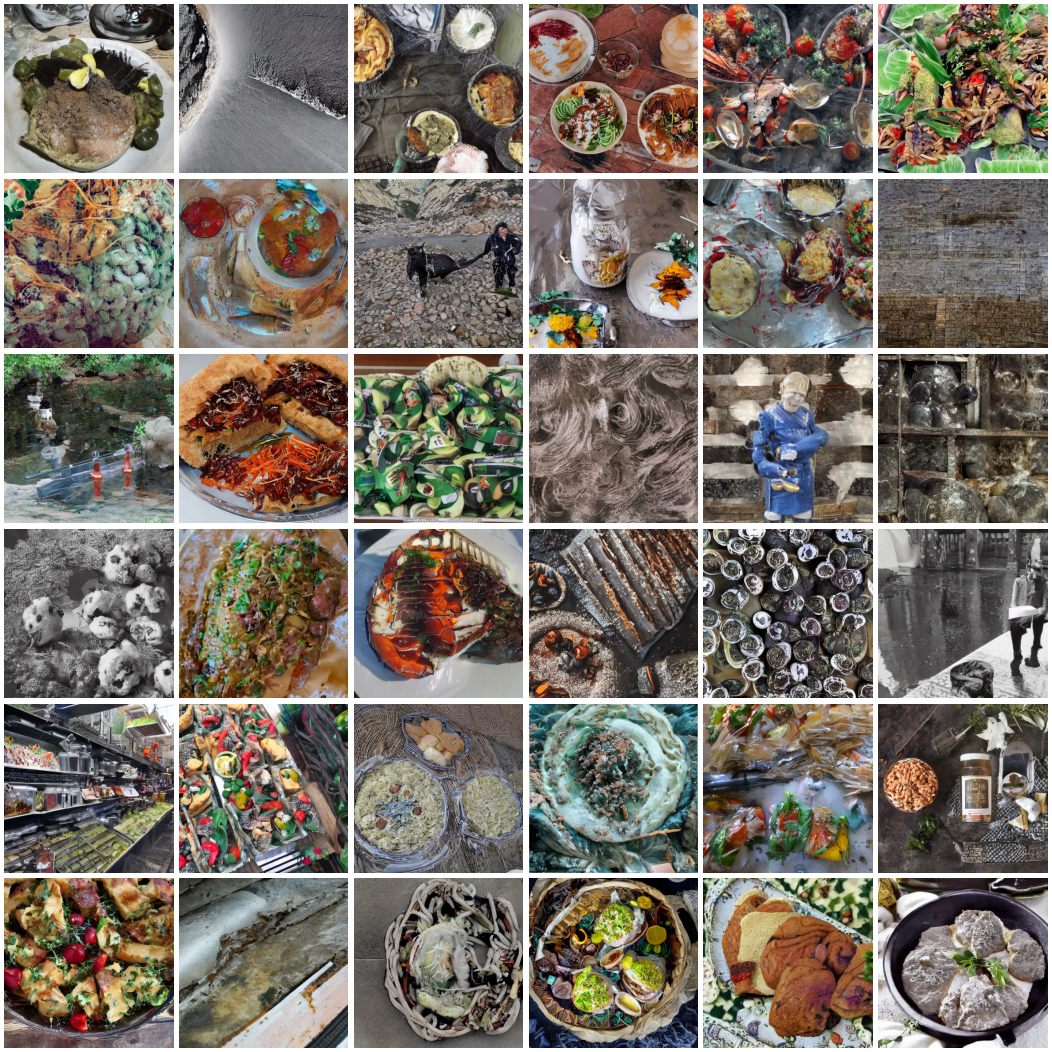

Figure 10: Good and bad example images generated by the diffusion model. The model took 30 denoising steps to generate the images.

Figure 11: Each row shows the input ingredients (left side) and the generated image (right side). We remove ingredients to condition on while keeping the random noise fixed to investigate the effect of the conditioning on the generated image.