Final Report: Image to Textual Recipe Generation

Ayush Agarwal [*] Dhruv Arya [*] Kushagra Mahajan [*] Rishubh Gupta [*] Youngmin Kim [*]

Abstract

In this project, we aim to work on the task of image to textual recipe generation. Informed from our previous analyses and past work done in the domain we investigate the task further. We explore new research ideas which have the potential of incrementally improving different stages of the recipe generation to produce better recipes and evaluation mechanisms. We also perform comparison between our approaches and their respective state-of-the-art baseline models through qualitative and quantitative analysis. We perform an in-depth discussion on the reasons for success or failure of the various methods, and try to propose methods to address these shortcomings. The code can be found at `https://github.com/kushagramahajan/MMML-Recipe-Generation`.

1. Introduction

Food carries great cultural importance in our lives. It is often an indicator of our health and culture. Different foods have varied ingredient compositions and cooking mechanisms. Components need to be added in an appropriate ratio and intricate cooking instructions need to be followed to get the desired taste. Determining all of these from just the food images and without any prior knowledge is an extremely challenging task. In our work, we explore the task of food image to recipe instruction generation using the Recipe1M dataset (Salvador et al., 2017). This work involves the detailed investigation of 4 different research ideas which attempt to explore the recipe generation task in greater depth with the goal of improving performance and the evaluation methodologies.

Solving this task can have useful applications where someone can take a photo and is able to know how to create that dish. Creating a model that can recognise ingredients, form a recipe from them and sequence them properly can be valuable for a common audience around the world.

Some of the technical challenges involved with this research problem are as follows. First, in case of a prepared dish it can be very hard to determine all the ingredients present without having some prior knowledge. This is because several ingredients may be in a different form/texture/color than their original form. Second, large image datasets and generative models can both be a highly computationally intensive task.

Title: Antipasto pasta salad

Ingredients: olive, pasta, cheese, pepper, onion, salami, artichoke, tomato, cucumber, salt, mayonnaise, vinegar, mustard.

Instructions:
- Cook pasta according to package directions.
- Drain and rinse with cold water.
- In a large bowl, combine pasta, salami, artichoke hearts, red onion, olives, tomatoes, and cheese.
- In a small bowl, whisk together mayonnaise, vinegar, mustard, salt, and pepper.
- Pour dressing over pasta mixture and toss to coat.
- Cover and refrigerate for at least 1 hour before serving.

Figure 1. Sample dataset recipe comprising of a title, ingredients and cooking instructions

Learning from these challenges, unimodal analyses, and the experiments on the baselines, we propose four research ideas with the aim to improve the overall recipe generation quality, reduce the amount of supervision and computation without significant impact on the performance, and switch to more realistic and useful evaluation metrics that capture the recipe correctness more effectively. Our contributions can be summarized as follows:

- Providing additional information apart from ingredients in the form of cooking tools and action verbs resulting in a more coherent recipe and sentence structure.
- Applying co-learning to reduce the dependence on ground truth ingredients during the training phase. By masking out a fixed percentage of the ingredient input during training, the resulting model is more robust against missing ingredients.
- Exploring the use of contrastive loss to force images of similar dishes to be closer in the embedding space and dissimilar dishes to be further apart, thereby improving the generated recipes.

- Improving the evaluation mechanism by proposing a core-component based evaluation metric, drawing on the fact that conventional metrics like BLEU or perplexity are not the most suitable for recipe formats along with the use of metrics like ROUGE and METEOR.

2. Related Work

Sener & Yao (2019) solved the task of recipe completion from incomplete input videos from the Tasty Videos Dataset. They use a seq2seq architecture to learn the prediction of the next recipe instruction where the encoder portion is interchangeable between a text encoder and a video encoder.

H. Lee et al. (2020) work on ingredient and instruction generation from recipe titles and cooking instructions. While they perform a generative task for recipes they do it in a text-to-text fashion. They use BLEU3, ROUGE4 as well as normalized tree edit distance as the evaluation metrics. They also build a recipe evaluation module that allows users to conveniently inspect the quality of the generated recipe contents. Salvador et al. (2019) generate recipes given the image with an intermediate step of predicting the ingredients from the image.

Several tasks using recipe datasets including the Recipe1M dataset leverage ingredient information to aid in the final objective. Marin et al. (2018), Jiang et al. (2020), Wang et al. (2020) and Salvador et al. (2019) all encode ingredient information into latent embeddings to solve various tasks including recipe generation, ingredient recognition, recipe/image retrieval. These approaches take advantage of the fact that ingredients are extractable from both the recipe and the image and can provide a more structured understanding of both modalities.

Based on the previous work in this domain and our baseline implementations we propose several ideas to improve the shortcomings of current approaches in the image to recipe generation architecture. In Section 3 we discuss our problem statement. Then in Section 4 we discuss approaches in further detail. We then talk about our experimental methodology in Section 5. We discuss the baseline results and detail our error analysis in Section 6. Finally we propose future research directions in Section 7.

3. Multimodal baseline models

The primary baseline model against which we compare all of research ideas is the paper on Inverse Cooking (Salvador et al., 2019). The paper uses ResNet-50 to encode the image and then uses a decoder architecture to predict ingredients, followed by a single embedding layer mapping each ingredient into a fixed-size vector. The instruction decoder, which is used to generate the final recipe, is composed of transformer blocks, each of them containing two attention layers followed by a linear layer. The first attention layer applies self-attention over previously generated outputs, whereas the second one attends to the model conditioning in order to refine the self-attention output. The transformer model is composed of multiple transformer blocks followed by a linear layer and a softmax non-linearity that provides a distribution over recipe words for each time step t.

The recipe generator is conditioned on two sources: the image features, and ingredient embeddings. To attend over multiple modalities, the paper proposes three types of fusion techniques — attention over the concatenated embeddings, unimodal attention followed by summation, and arranging all embeddings as one sequence and then attending over them. The paper shows that concatenated embeddings perform the best for this task.

The model is trained in two stages. First, a part of the model is trained for ingredient prediction by using the ground truth ingredients as the target. We call this the '**phase 1**' of training. Next, the model is trained end-to-end for recipe generation in what is referred to as '**phase 2**' of training. The model uses teacher forcing for all training tasks except when predicting the set of ingredients. Therefore, during the second training phase, ground truth ingredients are fed to the model.

The model has a different loss function for each stage. The objective used in phase 1 of the training is a combination of three different losses. The first loss is a set-based ingredient prediction loss. In order not to penalize for the order in which the ingredients were predicted, max-pooling is performed across all time steps and binary cross-entropy between the pooled predictions and ingredients is computed. The second loss is the *eos* loss, which is a binary cross entropy loss between the predicted *eos* probability at all time step and the ground truth. The third loss is a *l1* cardinality penalty.

For recipe generation, the model minimizes the negative log likelihood:

$$-\sum_{i=1}^{L} \log(p(y_i;\theta))$$

Here L is the length of the sequence, θ represents the model parameters, y_i is the true label at index i.

During the inference phase of ingredient generation, the ingredient decoder keeps generating tokens until an *eos* (end of sentence) token is generated. At each timestep, the ingredient with the highest activation is sampled. For recipe generation, tokens are sampled greedily until the *eos* token is encountered. Adam (Kingma & Ba, 2014) was used to optimize the model in the original implementation.

4. Proposed Approach

4.1. Component-aware image embeddings

4.1.1. Motivation

Recipes contain some common types of core components, such as ingredients, cooking tools and action words. While sentence generation is a complex task, it becomes a simpler task if the core components are given to the model. Given the components, the model only needs to be able to generate coherent sentences surrounding the component words. In other words, component information can have the effect of reducing the generation output space of coherent recipe sentences. Based on this intuition, we experiment with providing component information to the model during training.

This method is further motivated by our human recipe generation exercise. Prior to experimentation, we had 5 human agents write down a total of 30 sample recipes from the Recipe1M dataset given the image of the dish. The agents first wrote recipes after only viewing the image, then wrote another version of the same recipe after having looked at the ground truth ingredient list as well. We observed a significant spike in the BLEU score and the qualitative evaluation of the recipes when the agents had access to the ground truth ingredients. The average BLEU score before and after looking at the ingredients were 0.002 and 0.0186 respectively. When the human agents know which ingredients are supposed to appear in the recipe, they can use that information to infer what kind of actions would most likely be performed to those ingredients and what kind of tools might commonly be used to perform that action.

It is our hypothesis that if the models are told which core words (e.g. tools, verbs) are expected to appear in the recipe, it would be able to more easily infer what other words would appear alongside those core words. This would in turn enhance the quality of the model-generated recipes in an analogous manner to our observation from the human generation experiment.

4.1.2. Model Description

The implementation of this approach is based on our baseline model, Inverse Cooking. Inverse Cooking already has image to ingredient prediction as an intermediate step before instruction generation. We extend this architecture by adding cooking tool prediction and action word prediction. Figure 2 shows the implementation of our component-aware embedding extension architecture. The main difference with Inverse Cooking is two extra sets of image embeddings are fed into the instruction decoder, obtained from the tool encoder and the action encoder.

4.1.3. Learning

We follow the overall training process of Inverse Cooking as described in Section 3 to train our model. In phase 1 of the training, the objective function is the average of the binary cross-entropy of the ingredient, tool and action decoders' predictions. Phase 2 of the training is identical to that of Inverse Cooking.

4.2. Co-learning Ingredients from the Food Image Encoding

4.2.1. Motivation

We use the Inverse Cooking architecture as the baseline for this research idea. The objective of this research idea is to reduce the dependency on ingredients since they add additional supervision requirement for our models. This is particularly useful when we do not have ingredient information for a lot of samples during training. Our hypothesis was that reducing the ingredient supervision would not have a huge detrimental impact on the recipe generation quality given that our image encoder was trained with ingredient supervision, and we perform co-learning of the ingredients for recipe instruction decoding. We validate this hypothesis in a step-wise way to observe the impact of each step where we remove ingredient information. We also hypothesize that using ingredient as an input to the recipe decoder can lead to error propagation if the predicted ingredients are incorrect resulting in a poor recipe being generated. However, if no ingredient information is used as input in such scenarios, the recipe generated might be better. Also, co-learning is needed since feeding unimodal image encoding input to the inverse cooking model during test time results in extremely poor performance for recipe generation.

4.2.2. Model Description

We begin with the co-learning approach in which we remove the ingredient branch during test time. However, we do not completely remove ingredient based supervision during the training stage. We find that there is only a minor decrease in performance which is encouraging since it indicates that we are able to remove ingredient supervision to a large extent without too much impact on performance. The ideal scenario would be to completely remove ingredients even from the entire training pipeline. Thus, we modify the inverse cooking architecture to remove the two-phase approach and train everything in a single phase ie. direct image to recipe generation.

Here we only make use of the recipe supervision. We find that the performance reduces considerably. This is due to the fact that there is no prior training of the image encoder and it is being trained end-to-end as part of the entire pipeline. Some form of supervision is needed to train the im-

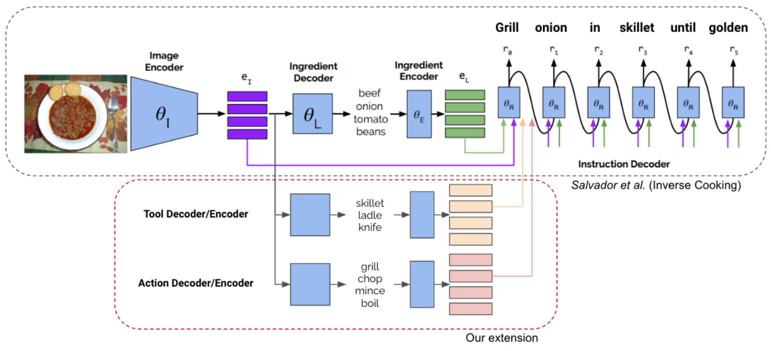


Figure 2. Inverse Cooking architecture and our Component-Aware Embedding extension

age encoder to obtain better perplexity scores for the recipe instruction generation task. So, we need some ingredient information for training phase 1 of inverse cooking as discussed in section 3. We take a subset of the training data for training the instruction decoder using the ingredient supervision signal, which trains the image encoder in the process. The remainder of the training data is used for training the instruction decoder without any ingredient information. Thus, we use varying amounts of ingredient information for training the instruction decoder, and determine the optimal number of samples where the ingredient information should be masked out for the best unimodal performance at test time considering only the image encoding.

During test time, we feed the model just the food dish image. This is similar to inverse cooking. However, the difference lies in the fact that inverse cooking feeds the image embeddings and the ingredient predictions as input to the instruction decoder. However, our model only feeds the image embeddings to the instruction decoder and does not use the predicted ingredients as input to the instruction decoder.

4.2.3. Learning

Details about the model architecture and training stages is the same as the baseline model as detailed in section 3. The model has a different loss function for each stage. In the ingredient prediction task, max-pooling is performed across all the time steps and binary cross-entropy between the pooled predictions and ingredients is used to train the ingredient predictor. For recipe generation, the model minimizes the negative log likelihood just as the baseline described in Section 3.

4.3. Exploiting Semantic Food Image Similarity for Recipe Generation

4.3.1. Motivation

As can be observed in the Figure 3, both images are of chocolate chip cookies however they look different based

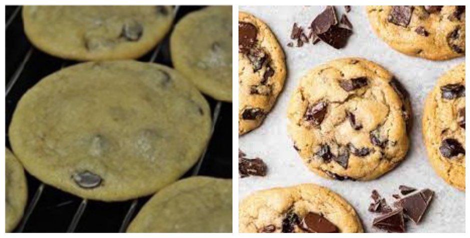


Figure 3. Images of chocolate chip cookies with different lighting and capture angle conditions

on the angle, size and lighting. Even though they belong to the same category, the embeddings generated from an image encoder for both of them are not as close as expected.

Recipe generated for the first image in Figure 3:

"Preheat oven to 350 degrees. Mix all dry ingredients in a medium bowl. In a separate bowl, cream butter and sugars until light, about 5 minutes. Add egg and vanilla and beat until combined. Gradually add dry ingredients into butter mixture and beat in well. Bake 8-10 minutes until golden."

Recipe generated for the second image in Figure 3:

"Preheat oven to 350 degrees f (175 degrees c). Grease cookie sheets. In a large bowl, cream together the butter and sugar until light and fluffy. Beat in the egg and vanilla. Stir in the flour, baking soda, and salt. Stir in the chocolate chips. Drop by rounded teaspoonfuls onto the prepared cookie sheets. Bake for 8 to 10 minutes in the preheated oven, or until lightly browned."

The first recipe misses out key ingredients such as flour and baking soda. It mentions the temperature of the oven as "350 degrees" but does not mention the unit Fahrenheit. The first recipe also misses out on the instruction "Grease the cooking sheets". Moreover, the second recipe is generally more comprehensive.

The goal of this idea is to use contrastive learning in order to bring the image embeddings of the images belonging to the same recipe type, such as ”chocolate chip cookie” closer together so that the recipe generation is improved for the overall dataset.

4.3.2. Model Description

Contrastive learning was applied on the image encoder of Inverse Cooking (Salvador et al., 2019). This was done using three different approaches. The first one is a supervised technique where the title of the recipes were use to bring image embeddings closer. This was done by projecting the title into the embedding space and finding the similarity in the titles and propogating the similarity to the image encoder.

The second approach uses clustering. The image titles were trimmed down to a basic form. This was done by removing the word ”Recipe” and the possessive nouns in the titles. For example ”Uncle Shawn’s Chocolate Chip Cookies Recipe” was trimmed down to ”Chocolate Chip Cookies”. Using these trimmed recipes, clusters were created. Self-Supervised Learning was used to train the image encoder to generate feature maps where the images clustered together are closer in the image embedding space.

The third is a self-supervised image representation learning approach called BYOL (Grill et al., 2020). Since the idea is to improve the embeddings of images with poor lighting or angles, the approach includes augmenting the image and training the model to generate better representations of the augmented image with the original image as the reference.

4.3.3. Learning

The image encoder used in Inverse Cooking is a Resnet-50 trained on ImageNet. The underlying idea to is to train the Resnet to generate better representations.

For the supervised technique, we use the VirTex (Desai & Johnson, 2021) model whose goal is to learn visual representations using a pretraining approach using the titles of recipes. The model has two components: a visual backbone and a textual head. The visual backbone extracts visual features from an input image I. The textual head accepts these features and predicts the title C = (c1, . . . , cT) token by token. The textual head performs bidirectional captioning to generate the titles. All model components are jointly trained to maximize the log-likelihood of the correct caption tokens where θ, ϕ_f, and ϕ_b are the parameters of the visual backbone, forward, and backward models respectively.

$$\mathcal{L}(\theta, \phi) = \textstyle\sum_{t=1}^{T+1} \log \left(p\left(c_t \mid c_{0:t-1}, I; \phi_f, \theta\right)\right)$$

$$+ \textstyle\sum_{t=0}^{T} \log \left(p\left(c_t \mid c_{t+1:T+1}, I; \phi_b, \theta\right)\right)$$

For the clustering approach, VISSL (Goyal et al., 2021) was used. The specific algorithm used was SimCLR (Chen et al., 2020) which is a simple framework for contrastive learning of visual representations. In the SimCLR technique, a minibatch is randomly sampled of size N and the images in the batch are augmented resulting in 2N data points. The other 2(N-1) samples in the minibatch are considered as negative samples. Then the loss function for a positive pair of examples (i, j) is defined as

$$\ell_{i,j} = -\log \frac{\exp\left(\operatorname{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right)/\tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp\left(\operatorname{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_k\right)/\tau\right)}$$

where $\mathbb{1}_{[k \neq i]} \in \{0, 1\}$ is an indicator function evaluating to 1 iff $k \neq i$ and τ denotes a temperature parameter.

For the self-supervised approach, BYOL (Grill et al., 2020) is used which relies on two neural networks, referred to as online and target networks, that interact and learn from each other. From an augmented view of an image, the model trains the online network to predict the target network representation of the same image under a different augmented view. At the same time, the target network is updated with a slow-moving average of the online network.

The mean squared error between the normalized predictions and target projections is calculated as follows:

$$\mathcal{L}_{\theta,\xi} \triangleq \left\|\overline{q_\theta}\left(z_\theta\right) - \bar{z}'_\xi\right\|_2^2 = 2 - 2 \cdot \frac{\left\langle q_\theta\left(z_\theta\right), z'_\xi\right\rangle}{\left\|q_\theta\left(z_\theta\right)\right\|_2 \cdot \left\|z'_\xi\right\|_2}$$

4.4. Improved Evaluation Metrics

4.4.1. Motivation

While evaluating manually written recipes we saw the shortcomings of using BLEU and the perplexity scores. BLEU score works by counting matching n-grams between the reference and the generated text. Perplexity is a way to capture the degree of uncertainty a model has in predicting (i.e. assigning probabilities to) text. This means that the lower the perplexity, the lower the uncertainty, and the higher the probability. For our task of recipe generation, these are poor strategies as the same recipe can be written in different ways, with rephrased sentences, and sentences being combined in different ways. BLEU score is not able to focus on more meaningful parts of information between the generated and the reference sentences such as ingredients for our example of recipe instructions. Further in the baseline analysis, we observed that our baseline models perform poorly at generating recipes with essential recipe elements like ingredients, tools and cooking verbs in the generated recipes and the current evaluation metrics do not focus on these essential recipe elements. Thus, there is a need for newer evaluation metrics which focus on more important components of a recipe rather than n-grams. We show the most frequent verbs and tools present in the dataset in Figure 4 and Figure 5

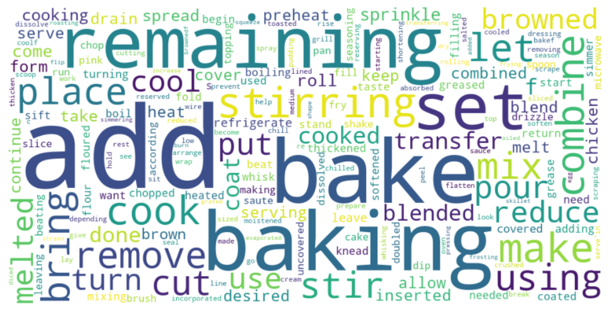


Figure 4. Word cloud for most frequent verbs

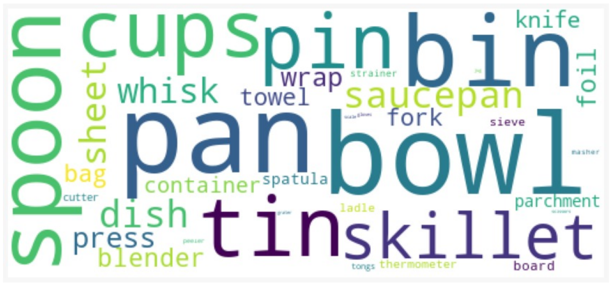


Figure 5. Word cloud for most frequent tools

Below are some of the evaluation metrics that we use in place of BLEU and Perplexity. Out of the below metric, POS Tagging Based Eval is proposed by us, while ROUGE and METEOR already exist in the literature. We find that all these 3 metrics are more suitable for much more suitable for measuring the similarity of the generated recipe instructions with the reference recipe instructions compared to BLEU and Perplexity. Each of these metrics focuses on a different aspect of generation for our task.

ROUGE
The ROUGE-L metric measures the longest matching sequence of words using Longest Common Subsequence (LCS) algorithm. Since LCS does not require consecutive matches but instead uses in-sequence matches that reflect sentence level word order, we believe ROUGE-L to be a better evaluation metric for recipe generation.

METEOR
METEOR is based on aligning the reference and generated sentences based on exact, stem, synonym, and paraphrase matches between words and phrases. Since METEOR relies on alignment rather than consecutive tokens, we believe it makes it a good metric for recipe generation.

POS Tagging Based Eval
This involves using the NLTK based POS tagger to extract the ingredients, tools, and verbs from the recipe instructions. A large overlap between the ingredients, tools and verbs of the reference and the generated recipes indicates higher similarity between the reference and generated recipes. The metric is particularly relevant for recipes since it focuses exclusively on components that are important for recipes.

4.4.2. LEARNING

For a food dish image img_t, we obtain the ground-truth or the reference recipe r_R and the generated recipe r_G. We then extract the ingredients i, tools t and verbs v from both the reference and the generated recipes. The process for the extraction of these components has been described in detail in section 5.3.4. We then perform an intersection over union (IoU) for each of the three components present in r_R and r_G. The IoUs are aggregated for each of the three components by extracting the mean IoU for each component. We also provide the overall IoU by averaging across the three component IoUs.

5. Experimental Setup

5.1. Dataset and Input Modalities

All experimentation presented in this paper is conducted on the Recipe1M (Salvador et al., 2017) dataset, which is a large-scale, structured corpus of around one million textual recipes recipes and around 900K food dish images. It is collected from over two dozen popular cooking websites, and includes English cooking recipes (ingredients and instructions), recipe titles, images, and categories. Since the dataset was obtained by scraping cooking websites, the original dataset which had 16,823 unique ingredients had to preprocessed to reduce the ingredient vocabulary to 1,488 unique ingredients. Keeping in consideration our compute limitations, we use only the validation dataset for all our experimentation comprising 155K recipes. The validation data is split into training, validation, and test subsets in a 50:25:25 ratio.

5.2. Multimodal baseline models

The baseline model has been discussed in Section 3. Please refer to it for a detailed explanation.

5.3. Experimental methodology

The experiments conducted for each of the research ideas are highlighted below:

5.3.1. COMPONENT-AWARE IMAGE EMBEDDINGS

In order to add tool and action decoders to the model, we first generated a vocabulary of tool and action words that are found in the dataset. The vocabularies used in this experiment are the same vocabularies obtained through the process described in 5.3.4.

The ground truth set of tools and actions that are generated are used in phase 1 of the training. Each component decoder is trained to predict the ground truth set of components given the image.

In phase 2 of the training, the image is passed through all three of the component decoder/encoder pipelines. The three types of resulting component-aware image embeddings - ingredient-aware, tool-aware and action-aware - are concatenated to raw image embeddings and then fed into the instruction decoder, which generates instruction sentences.

5.3.2. Co-learning Ingredients from the Food Image Encoding

We conducted experiments to showcase the effectiveness of the co-learning approach. To begin with, we split the training data into 2 disjoint subsets comprising sizes 20% and 80%. The smaller subset was used for training phase 1 of the ingredient decoder. The image encoder is trained in the process. The larger subset was used for training the recipe instruction decoder using only a fraction of the samples with ingredient information and the rest of them with masked out ingredient information. This is done to co-learn the ingredients such that we do not need to pass the ingredient information to the recipe instruction decoder during the testing phase.

We perform experiments to show why co-learning is essential to obtain a good recipe generation performance when using unimodal data, and to determine the optimal threshold for number of unimodal samples for co-learning, we experiment with training the recipe instruction decoder for different percentages of unimodal samples.

5.3.3. Exploiting Semantic Food Image Similarity for Recipe Generation

We conducted three experiments to understand the effect of Contrastive Learning on the image embeddings. The first experiment uses the VirTex (Desai & Johnson, 2021) model to train the image encoder to generate recipe titles. The model is trained on the training split we are using to train the Inverse Cooking model. The VirTex model converges around 100,000 iterations with the default parameters. The model has a linear layer which is removed and then replaced with a new linear layer which is then trained during Inverse Cooking's **Phase 1** training.

The second experiment uses clustering. The titles are reduced to a basic form and the recipes are clustered based on the basic form of the titles. Using the SimCLR (Chen et al., 2020) method, the images are sampled based on which cluster they belong to. The batch has a sample from each cluster such that each of the entries in the batch is a negative sample for an each. The model takes 33 epochs to converge with the default parameters. We use the backbone of the Resnet trained using SimCLR and add a linear layer which is then trained during Inverse Cooking's **Phase 1** training.

The third experiment required pre-processing the dataset. Since the idea was to understand the impact of a Self-Supervised Learning, we split the training set into 70-30 where 70 percent of the training set was used to train the SSL training ad 30 percent of the data was used to train the Inverse Cooking model. The 70 percent is used to train a Resnet-50 using the approach called BYOL (Grill et al., 2020). The model takes 3 epochs to converge with the default parameters. The Resnet-50 is now plugged into Inverse Cooking which is trained on remaining 30 percent of the data.

5.3.4. Improved Evaluation Metrics

We use the Part of Speech (POS) Tagging model created by NLTK to identify verbs and nouns present in the training set of our ground truth recipes. We create a set of tools and verbs by filtering out the nouns and verbs tagged by the model. We obtain the ingredients through the metadata provided with the Recipe1M dataset. We then use the sets of tools, verbs and ingredients for calculating the POS Tagging based Eval metric for the reference and generated recipes for each food image as described in section 4.4.

Further for each image we calculate BLEU, METEOR and ROUGE scores for the pair of reference and generated recipes. We present our results for all metrics and perform a detailed analysis in section 6.4

6. Results and Discussion

6.1. Component-aware image embeddings

	Baseline	Comp-Aware
BLEU	0.0190	**0.0283**
ROUGE-L	0.1821	**0.1963**
METEOR	0.1602	**0.2285**
Perplexity	13.95	**12.65**

Table 1. Comparison of BLEU, ROUGE-L, METEOR and Perplexity.

We observe that adding component-aware embeddings results in an improvement in the overall quality of the generated recipes. Table 1 shows an improvement on all metrics measured in comparison to the Inverse Cooking baseline.

We also observe improvements in the component-specific IoU metrics as well. This is measured from the generated recipes, and is thus different from the component IoU measured on the component decoders. Table 5 shows the component IoU measured from the generated recipes. As ingredients, tools and action verbs constitute the core of a recipe, it is important that the model is proficient in retrieving these core components.

An improvement in the tool and verb scores is expected, given that only our model is exposed to the ground truth tools and verbs during training. It is however interesting to note that ingredient IoU also improved, even though both models are exposed to the same ground truth ingredients during training. This suggests that the additional information about the verbs and tools also helps the instruction decoder in identifying the correct ingredients that should occur in the recipe. From this, we can further infer that the aggregate contribution of the three component decoders is more than a simple sum of the contributions from the individual decoders.

6.2. Co-learning Ingredients from the Food Image Encoding

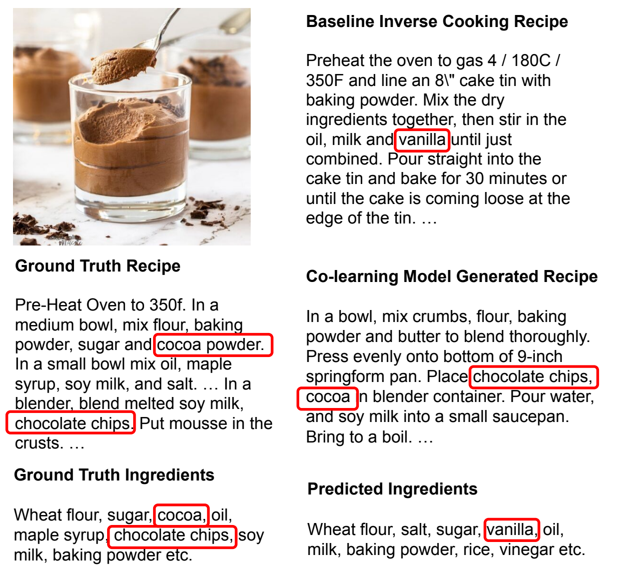
We find that without using ingredients in recipe generator training and testing, we obtain a test perplexity of 15.82 in contrast to a perplexity value of 13.95 (inverse cooking baseline) which is obtained with ingredients. If we completely remove ingredients from the training and testing stages, ie. train directly from image to recipe generation, we obtain a really bad performance. Our instruction perplexity increases to 23 indicating that direct image to recipe generation is a very complex task due to the complicated structure and ordering of recipes, and thus we need some intermediate supervision to make the task tractable. This is achieved through the use of supervision for ingredient decoder training on 20% of the data. Table 2 shows the decrease in performance (1.8 perplexity points) with varied levels of ingredient supervision used during the instruction decoding stage (ie. stage 2).

	Percentage of missing ingredients		
Dataset split	30%	50%	70%
val	16.2787	**16.1783**	16.4559
test	16.0348	**15.8247**	15.9854

Table 2. Perplexity for different proportions of missing ingredients in training the recipe instruction decoder

We see from Table 2 that the best test perplexity is obtained when 50% of the samples used for training the instruction decoder do not contain ingredients. Using a higher percentage unimodal data for co-learning leads to insufficient samples for the model to co-learn the ingredient information from the image encoding, while lower amount of unimodal data leads is insufficient to make the model robust the the missing modality.

To validate our hypothesis that co-learning improves robustness to lack of one modality, we test the baseline inverse cooking model by omitting the ingredient modality during the recipe instruction decoding process. This results in a recipe perplexity value of 30.44, compared to a value of 13.95 with the ingredient modality. However, when the ingredients are co-learned, we obtain the recipe perplexity value of 15.82 which is a considerable improvement.



Baseline Inverse Cooking Recipe

Preheat the oven to gas 4 / 180C / 350F and line an 8\" cake tin with baking powder. Mix the dry ingredients together, then stir in the oil, milk and vanilla until just combined. Pour straight into the cake tin and bake for 30 minutes or until the cake is coming loose at the edge of the tin. …

Ground Truth Recipe

Pre-Heat Oven to 350f. In a medium bowl, mix flour, baking powder, sugar and cocoa powder. In a small bowl mix oil, maple syrup, soy milk, and salt. … In a blender, blend melted soy milk, chocolate chips. Put mousse in the crusts. …

Co-learning Model Generated Recipe

In a bowl, mix crumbs, flour, baking powder and butter to blend thoroughly. Press evenly onto bottom of 9-inch springform pan. Place chocolate chips, cocoa in blender container. Pour water, and soy milk into a small saucepan. Bring to a boil. …

Ground Truth Ingredients

Wheat flour, sugar, cocoa, oil, maple syrup, chocolate chips, soy milk, baking powder etc.

Predicted Ingredients

Wheat flour, salt, sugar, vanilla, oil, milk, baking powder, rice, vinegar etc.

Figure 6. An example of preventing error propagation from ingredient prediction to recipe generation using our co-learning approach.

Another interesting observation was that when the decoded ingredients were incorrect, the recipe produced by the baseline inverse cooking approach was worse than what our approach produced. This is because incorrect ingredients send a wrong input signal to the instruction decoder resulting in a worse output. Figure 6 shows an example of such a case. For the chocolate mousse shown in the figure, the ingredients ‘cocoa’ and ‘chocolate chips’ are not predicted. In its place, the ingredient ‘vanilla’ is predicted. Thus, the inverse cooking baseline model which uses these ingredients in the testing phase generated the recipe of a vanilla mousse. However, using our co-learning based research idea, since we are not feeding the predicted ingredients as input to the recipe decoder during test time, the recipe predicted is that of a chocolate mousse itself. Thus co-learning of the ingredients makes the model robust to noisy ingredient predictions.

Model	Perplexity
100% Data Baseline	13.95
100% Data TitlePrediction	14.4

Table 3. Comparison of different image encoder pretraining approaches

Model	Perplexity
Baseline	15.09
DataAugmentationBasedContrastive	13.97
TitleClusteringBasedContrastive	14.28

Table 4. Comparison of different self-supervised image encoder pretraining approaches with the baseline when only 30% data is available for supervised learning

6.3. Exploiting Semantic Food Image Similarity for Recipe Generation

We find that pretraining an image encoder using a title generation task using all of the training data does not result in improved performance over the baseline inversecooking approach (Table 3). We observe an increase in perplexity by around 0.5 which indicates that the image encoder is worse than the one trained using a classification objective on ImageNet. Since in this experiment, the image encoder is not frozen during phase 1 training, an encoder comparable to the baseline should receive enough training signal for the overall perplexity to be similar. However, the higher perplexity indicates that the image encoder trained using the title generation task is not suitable for recipe generation.

Table 4 shows the results of using contrastive learning to train the image encoder with 70% of the training data and then using the remaining 30% data for the supervised recipe generation training. We observe that both the contrastive learning approaches generate image encoders that are better than the baseline approach with respect to the recipe generation perplexity. It must be noted that these image encoders are only trained during the contrastive learning phase and are frozen during the supervised recipe generation training. Thus, the only signal for training these encoders comes from the contrastive learning objective. It can be seen that the data augmentation-based contrastive learning objective leads to a lower perplexity. This is expected because the initial error analysis had revealed that existing models don't perform well when the image quality is reduced. This training objective explicitly trains the encoder to map lower quality versions of images to the same space as the high quality versions.

A qualitative comparison of recipe generations between the best contrastive learning approach (DataAugmentationBasedContrastive) and the baseline reveals that the proposed approach performs much better when the input image is of poor quality. Figure 7 shows one such example. While the baseline model generates a completely incorrect recipe for the blurry image of chocolate chip cookies, the proposed approach is able to identify the dish correctly and is able to generate the right recipe. Thus, contrastive learning-based approaches can generate more suitable image encoders even when there is no supervised learning signal during recipe generation.

Model: Baseline
Recipe:(Incorrect)
Title: Vanilla pudding ii
Instructions:
-In a large saucepan, stir together the sugar, milk, butter, flour, salt, eggs, vanilla extract, and eggs.
-Bring to a boil …
…
-In a large bowl, combine the eggs, milk and vanilla …

Model: DataAugmentationBasedContrastive
Recipe: (Correct)
Title: Chocolate chip cookies
Instructions:
-Preheat oven to 350 degrees f (175 degrees c).
-In a large bowl, cream together the butter and sugar until light and fluffy.
…
-Stir in the chocolate chips.
-Drop by rounded teaspoonfuls onto ungreased cookie sheets.
-Bake for 8 to 10 minutes in the preheated oven, or until lightly browned.
-Cool on wire racks.

Figure 7. The contrastively learned image encoder-based model produces much better recipes for images that are blurry or poorly lit. Full recipes in Figure 13

6.4. Improved Evaluation Metrics

In this section we first discuss results with our proposed metrics along with BLEU, ROUGE METEOR for each of our research ideas in compassion with the baseline model. We then discuss the qualitative analysis and the merits of proposed evaluation metric in the the Discussion section.

6.4.1. Results

We first discuss the results with our proposed evaluation metric for each of our approaches in Table 5. We describe the Ingredient Score, Tool Score, Verb Score and the CompScore which is an aggregate of Ingredient, Tool and Verb Score. Baseline refers to results calculated for the model in Section 3. Similarly, CoLearn refers to Section 4.2, ExSem refers to Section 4.3 and CompAware refers to Section 4.1.

We observe that for CompAware the proposed metrics show a distinct improvement over each component as compared to the baseline. This confirmed our previous analysis of CompAware which focuses on improving Component Recognition in the architecture

We also report BLEU, ROUGE and METEOR and the baseline model or each of our approaches in Table 6

	Ingredient	Tool	Verb	CompScore
Baseline	0.1984	0.2129	0.2129	0.2080
CoLearn	0.2158	0.2216	0.2068	0.2147
ExSem	0.1836	0.1854	0.1740	0.1810
CompAware	**0.2250**	**0.2463**	**0.2562**	**0.2425**

Table 5. Comparison of IOU Scores and CompScore for our research approaches

	BLEU	ROUGE-L	METEOR
Baseline	0.0190	0.1821	0.1602
CoLearn	0.0179	0.1859	0.1566
ExSem	0.0142	0.1704	0.1507
CompAware	**0.0283**	**0.1963**	**0.2285**

Table 6. Comparison of BLEU, ROUGE-L and METEOR metrics for our research approaches

We compare Tool Score, Verb Score and Ingredient Score in graphs 8,9 and 10 for baseline and our research ideas.

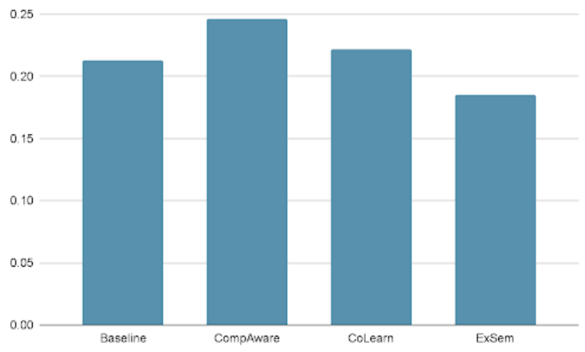


Figure 8. Comparison of Tool Score for Baseline, CompAware, CoLearn and Exsem

6.4.2. Discussion

We observe that the recipes getting a higher CompScore are more informative. They not only contain more ingredients, verbs and tools as expected but the recipe as a whole is more logical and practical in terms of using it to prepare the food. We observed that several recipes with a higher CompScore had a lower BLEU, ROUGE and METEOR score. We saw several reasons for that namely that the number of steps in generated and ground truth recipe are different and the choice of words used (except the components is different).

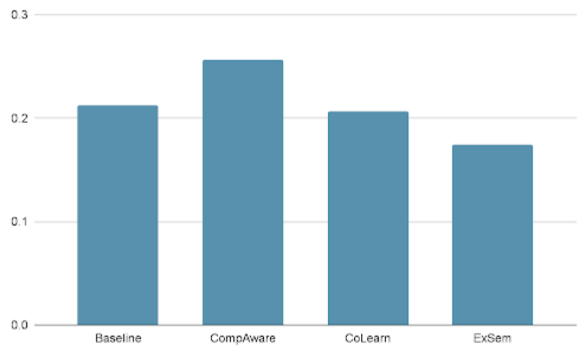


Figure 9. Comparison of Verb Score for Baseline, CompAware, CoLearn and Exsem

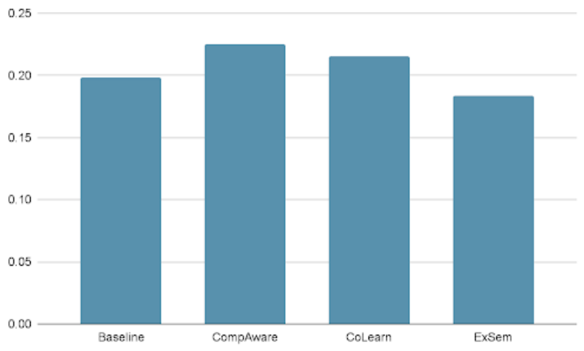


Figure 10. Comparison of Ingredient Score for Baseline, CompAware, CoLearn and Exsem

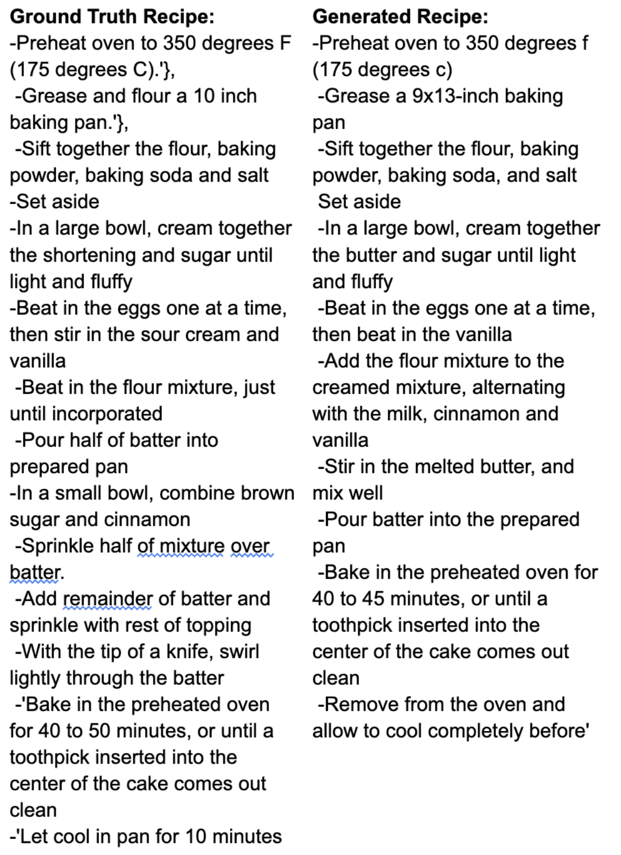
We discuss one such example in detail in Figure 11.

In Figure 11, for the generated recipe we get a CompScore of 0.69 whereas the BLEU score is 0.02. Upon analyzing the generated recipe we see that it is logically very similar to the ground truth recipe and includes all the essential components as well. This and several other examples in such cases show that generated recipes with higher CompScore are better in quality even when BLEU and other standard metrics like ROUGE and METEOR are low.

We also observe several cases where the CompScore is low but the BLEU score is very high. We discuss one such example in detail in Figure 12.

In Figure 12, for the generated recipe we get BLEU score 0.4678 but we get a low CompScore especially the Ingredient component. Upon analyzing the generated recipe we see while it is a very close match to the reference it has several ingredients like cinnamon, baking powder, baking soda missing and several ingredients like coconut, bundt, wafers and tools like tube and rack.

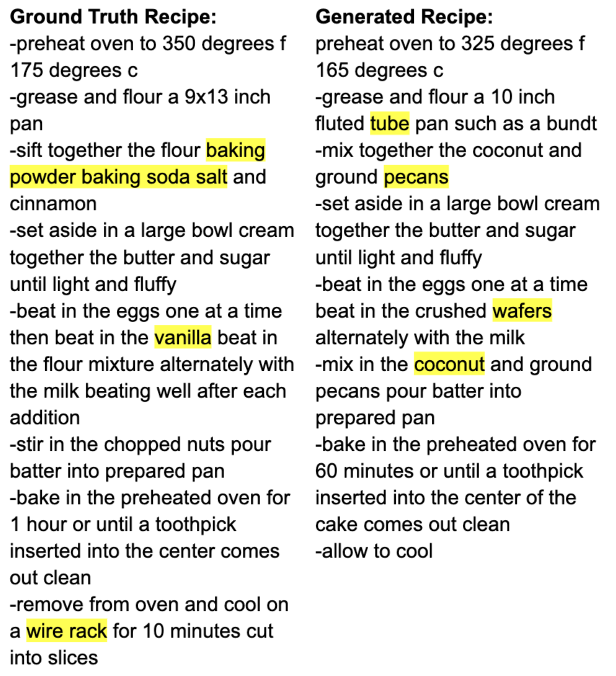
We discuss the quantitative results in the Results section



Ground Truth Recipe:
-Preheat oven to 350 degrees F (175 degrees C).'},
-Grease and flour a 10 inch baking pan.'},
-Sift together the flour, baking powder, baking soda and salt
-Set aside
-In a large bowl, cream together the shortening and sugar until light and fluffy
-Beat in the eggs one at a time, then stir in the sour cream and vanilla
-Beat in the flour mixture, just until incorporated
-Pour half of batter into prepared pan
-In a small bowl, combine brown sugar and cinnamon
-Sprinkle half of mixture over batter.
-Add remainder of batter and sprinkle with rest of topping
-With the tip of a knife, swirl lightly through the batter
-'Bake in the preheated oven for 40 to 50 minutes, or until a toothpick inserted into the center of the cake comes out clean
-'Let cool in pan for 10 minutes

Generated Recipe:
-Preheat oven to 350 degrees f (175 degrees c)
-Grease a 9x13-inch baking pan
-Sift together the flour, baking powder, baking soda, and salt
Set aside
-In a large bowl, cream together the butter and sugar until light and fluffy
-Beat in the eggs one at a time, then beat in the vanilla
-Add the flour mixture to the creamed mixture, alternating with the milk, cinnamon and vanilla
-Stir in the melted butter, and mix well
-Pour batter into the prepared pan
-Bake in the preheated oven for 40 to 45 minutes, or until a toothpick inserted into the center of the cake comes out clean
-Remove from the oven and allow to cool completely before'

Figure 11. Example of Generation for Low BLEU and High Comp-Score



Ground Truth Recipe:
-preheat oven to 350 degrees f 175 degrees c
-grease and flour a 9x13 inch pan
-sift together the flour baking powder baking soda salt and cinnamon
-set aside in a large bowl cream together the butter and sugar until light and fluffy
-beat in the eggs one at a time then beat in the vanilla beat in the flour mixture alternately with the milk beating well after each addition
-stir in the chopped nuts pour batter into prepared pan
-bake in the preheated oven for 1 hour or until a toothpick inserted into the center comes out clean
-remove from oven and cool on a wire rack for 10 minutes cut into slices

Generated Recipe:
preheat oven to 325 degrees f 165 degrees c
-grease and flour a 10 inch fluted tube pan such as a bundt
-mix together the coconut and ground pecans
-set aside in a large bowl cream together the butter and sugar until light and fluffy
-beat in the eggs one at a time beat in the crushed wafers alternately with the milk
-mix in the coconut and ground pecans pour batter into prepared pan
-bake in the preheated oven for 60 minutes or until a toothpick inserted into the center of the cake comes out clean
-allow to cool

Figure 12. Example of Generation for High BLEU and Low Comp-Score

and discuss our Qualitative Analysis in this section. We observe that our proposed metrics are a better indicator for the quality of the recipe and focus on evaluating the essential components of a generated recipe.

7. Conclusion and Future Directions

Through the component-aware image embedding experiment, we were able to observe that additional information about core components of a recipe helps the instruction decoder generate higher-quality recipes. We also note that the addition of the two additional component decoders resulted in better ingredient retrieval as well, suggesting that the aggregate contribution of all three component decoders exceeds the sum of their individual contributions. The co-learning approach is able to reduce the ingredient supervision requirement while not having a significant impact on the recipe generation quality. It allows us to get rid of the ingredients modality during test phase. This is particularly helpful when the ingredients predicted are incorrect since it prevents error propagation, ultimately resulting in the generation of better recipe instructions. We also observe that contrastive learning approaches can be used to train better image encoders for recipe generation when the amount of labelled data is very less. For the proposed evaluation metric we see that it is a much better indicator for evaluating generated recipe and its quality than started metrics that looks at n-gram pairs

For future work, we intend to work on ways to improve direct image to recipe generation, drawing parallels with the state-of-the-art research in image captioning. We also want to focus on improving fine-grained food image recognition since this will likely improve recipe generation. This is motivated by the fact that currently there are several cases where a food dish is confused for a similar looking different dish, and thus the recipe generated is different from the reference. One of the approaches to explore this is to use CLIP model as the pre-trained model for the image encoder and build our research ideas on top of that.

References

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Desai, K. and Johnson, J. Virtex: Learning visual representations from textual annotations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11162–11173, 2021.

Goyal, P., Duval, Q., Reizenstein, J., Leavitt, M., Xu, M., Lefaudeux, B., Singh, M., Reis, V., Caron, M., Bojanowski, P., Joulin, A., and Misra, I. Vissl. https://github.com/facebookresearch/vissl, 2021.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

H. Lee, H., Shu, K., Achananuparp, P., Prasetyo, P. K., Liu, Y., Lim, E.-P., and Varshney, L. R. Recipegpt: Generative pre-training based cooking recipe generation and evaluation system. In *Companion Proceedings of the Web Conference 2020*, pp. 181–184, 2020.

Jiang, S., Min, W., Lyu, Y., and Liu, L. Few-shot food recognition via multi-view representation learning. *ACM Trans. Multimedia Comput. Commun. Appl.*, 16(3), jul 2020. ISSN 1551-6857. doi: 10.1145/3391624.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Marin, J., Biswas, A., Ofli, F., Hynes, N., Salvador, A., Aytar, Y., Weber, I., and Torralba, A. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images, 2018.

Salvador, A., Hynes, N., Aytar, Y., Marin, J., Ofli, F., Weber, I., and Torralba, A. Learning cross-modal embeddings for cooking recipes and food images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3020–3028, 2017.

Salvador, A., Drozdzal, M., Giró-i Nieto, X., and Romero, A. Inverse cooking: Recipe generation from food images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10453–10462, 2019.

Sener, F. and Yao, A. Zero-shot anticipation for instructional activities. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 862–871, 2019.

Wang, H., Lin, G., Hoi, S. C. H., and Miao, C. Structure-aware generation network for recipe generation from images, 2020.

8. Appendix

Model: Baseline
Recipe:(Incorrect)
Title: Vanilla pudding ii
Instructions:
-In a large saucepan, stir together the sugar, milk, butter, flour, salt, eggs, vanilla extract, and eggs.
-Bring to a boil over high heat.
-Reduce the heat to medium-low, coverand cook, stirring constantly, until thickened, about 15 minutes.
-Remove pan from heat, and cool.
-In a large bowl, combine the eggs, milk and vanilla.
-Beat until smooth.

Model: DataAugmentationBasedContrastive
Recipe: (Correct)
Title: Chocolate chip cookies
Instructions:
-Preheat oven to 350 degrees f (175 degrees c).
-In a large bowl, cream together the butter and sugar until light and fluffy.
-Beat in the eggs one at a time, then stir in the vanilla.
-In a separate bowl, sift together the flour, baking powder, baking soda and salt.
-Stir into the creamed mixture alternately with the milk, beating well after each addition.
-Stir in the chocolate chips.
-Drop by rounded teaspoonfuls onto ungreased cookie sheets.
-Bake for 8 to 10 minutes in the preheated oven, or until lightly browned.
-Cool on wire racks.

Figure 13. The contrastively learned image encoder-based model produces much better recipes for images that are blurry or poorly lit