FIRE: Food Image to REcipe generation

Prateek Chhikara\textsuperscript{1,2}, Dhiraj Chaurasia\textsuperscript{1,2}, Yifan Jiang\textsuperscript{1,2}, Omkar Masur\textsuperscript{1}, and Filip Ilievski\textsuperscript{2,3}

\textsuperscript{1}University of Southern California, USA, \textsuperscript{2}Information Sciences Institute, USA, \textsuperscript{3}Vrije Universiteit Amsterdam, Netherlands

{pchhikar, chaurasi, yifjia}@isi.edu, omasur@usc.edu, f.ilievski@vu.nl

Abstract

Food computing has emerged as a prominent multidisciplinary field of research in recent years. An ambitious goal of food computing is to develop end-to-end intelligent systems capable of autonomously producing recipe information for a food image. Current image-to-recipe methods are retrieval-based and their success depends heavily on the dataset size and diversity, as well as the quality of learned embeddings. Meanwhile, the emergence of powerful attention-based vision and language models presents a promising avenue for accurate and generalizable recipe generation, which has yet to be extensively explored. This paper proposes FIRE, a novel multimodal methodology tailored to recipe generation in the food computing domain, which generates the food title, ingredients, and cooking instructions based on input food images. FIRE leverages the BLIP model to generate titles, utilizes a Vision Transformer with a decoder for ingredient extraction, and employs the T5 model to generate recipes incorporating titles and ingredients as inputs. We showcase two practical applications that can benefit from integrating FIRE with large language model prompting: recipe customization to fit recipes to user preferences and recipe-to-code transformation to enable automated cooking processes. Our experimental findings validate the efficacy of our proposed approach, underscoring its potential for future advancements and widespread adoption in food computing.

1. Introduction

Food is not only a vital source of sustenance but also an integral part of our cultural identity, defining our lifestyle, traditions, and social interactions [31]. As the well-known saying goes, “Tell me what you eat, and I will tell you who you are.” [28] emphasizing the idea that an individual’s dietary choices reflect their identity [30]. Moreover, a person’s physical appearance and cognitive abilities often bear evidence of their dietary habits, as the selection of nutritious food contributes to the overall well-being of both the body and mind [46]. The advent of social media enables anyone to share captivating visuals of personal experiences related to the delectable food they consume. A simple search for hashtags like #food or #foodie yields millions of posts, underscoring the immense value of food in our society [17]. The significance of food accompanied by its large amounts of publicly available data has inspired food computing tasks [31] that associate visual depictions of dishes with symbolic information. An ambitious goal of food computing is to produce the recipe for a given food image, with applications such as food recommendation according to user preferences, recipe customization to accommodate cultural or religious factors, and automating cooking execution for higher efficiency and precision [33].

Generating detailed recipe information or cooking procedures solely from a food image presents a considerable challenge [40, 41]. Food computing has been of interest to the computer vision (CV) community, whose efforts to use image processing for food quality assurance can be traced back to 1996 [11]. State-of-the-art food image processing methods [15, 27, 41] use deep learning techniques to extract ingredients from images with limited success. Meanwhile, a popular natural language processing (NLP) application has been recipe generation, a procedural task of creating recipes based on a flexible set of ingredients as inputs. Typical models for recipe generation include [29, 41, 51]. We note that prior work has not connected the dots between the CV and NLP research in order to provide an end-to-end system that generates recipes from images. Moreover, current methods for food computing have not caught up with the most recent advances in NLP and CV, featuring diffusion

Cooking Instructions:

- Preheat oven to 350 degrees Fahrenheit.
- Spray pan with non-stick cooking spray.
- Heat milk, water, and butter to boiling.
- Stir in contents of both pouches of potatoes.
- Let stand one minute. Stir in corn.
- Spoon half the potato mixture in pan.
- Sprinkle half each of cheese and onions.
- Top with remaining potatoes.
- Sprinkle with remaining cheese and onions.
- Bake 10 to 15 minutes until cheese is melted.

Figure 1. Given a potentially unseen image, our method FIRE generates a corresponding recipe consisting of a title, ingredients, and cooking instructions.
models and language modeling.

This paper presents a novel multimodal methodology that we call FIRE (Food Image to REcipe generation). FIRE is designed to generate comprehensive recipe output for the food computing domain, including food titles, ingredients, and cooking instructions, based on input food images as shown in Figure 1. We leverage recent advancements in CV and language modeling to employ state-of-the-art (SotA) techniques that have demonstrated exceptional performance in various vision and language tasks. FIRE connects the dots between SotA models, using BLIP [21] for title generation, a Vision Transformer [9] with a decoder for ingredient extraction, and the T5 [37] model for cooking instruction generation. Furthermore, we highlight two practical applications that can benefit from integrating FIRE with prompting large language models: recipe customization for personalized recipe adaptation, and recipe-to-code generation, enabling automated cooking processes. The contributions of the paper are as follows:

1. We leverage the capabilities of Vision Transformers (ViT) [9] to get expressive embeddings from food images, which are subsequently fed into an attention-based decoder to extract the ingredients of the recipe.
2. We present an end-to-end pipeline for generating recipe titles and cooking instructions, utilizing SotA vision (BLIP) and language (T5) models, respectively.
3. Our multimodal approach outperforms the existing work based on two evaluation metrics: (a) set metrics for ingredient extraction and (b) document-level metrics for cooking instruction generation.
4. We showcase the ability of FIRE to support two novel food computing applications: Recipe Customization and Recipe to Code Generation, through integration with few-shot prompting of large LMs.

We organize this paper as follows; Section 2 of the paper gives a detailed overview of the related work in the field of food computing and its gap against SotA models. We describe our proposed methodology (FIRE) in Section 3. Section 4 describes the experimental setup we follow to obtain the results, which are presented in Section 5. Section 6 illustrates two advanced applications that can benefit from our proposed approach. Finally, we conclude our paper in Section 7 with future research directions. We make all of our code available to stimulate work on recipe generation.

2. Related Work

2.1. Food Computing

Recently, the importance of food and the availability of extensive multimodal food datasets, such as Food-101 [4], Recipe1M [26], and Recipes242k [39], have enabled computational research on food computing tasks [31]. We review prior work on food recognition and recipe generation.

Food Recognition is an image-to-text task requiring models to detect food categories in a food image. Recognition of food items can offer people comprehensive information and a better understanding of unfamiliar dishes, thereby improving other food-related applications as well [1, 34]. Previous works focus on extracting deep representations of food [15, 27, 41, 55]. Martinel et al. [27] adapted a slice convolution block in the residual network to capture features in images. Salvador et al. [41] proposed InverseCooking, an encoder-decoder framework to output the title of the food. Wang et al. [50] also utilized images to get the recipe by treating it as an image captioning task. Notably, earlier architectures are constrained as they tend to emphasize global features rather than local features and can not detect the ingredients overlapping in the image [19]. Instead, we employ a SotA vision encoder, ViT [9], to enhance the extraction of the local semantic segmentation.

Recipe Generation is a more complex text-to-sequence task generating food recipes based on the ingredients provided. To solve this task, models must possess knowledge of food composition, ingredients, and cooking procedures to perform the task accurately. Early attempts at recipe generation were constrained by limited model capacity and structure, leading to solutions that relied on information retrieval techniques [52, 56]. Wang et al. [52] developed a novel similarity and filtering algorithm to increase the search accuracy. Xie et al. [56], leveraged the cooking flow and eating features with other domain knowledge to enhance the searching process. More recent work relies on encoder-decoder structures to generate recipes [41, 51] with multimodal settings. Salvador et al. [41] presented a framework that utilizes encoded image and ingredients representations in recipe generation. Wang et al. [51] added tree structures within the encoder-decoder process to incorporate structure-level information. In contrast to prior unimodal work, our approach uses images as input and generates titles and ingredients as an intermediate representation, and uses them to generate recipes.

2.2. State-of-the-Art Models

We review state-of-the-art models that have not found a broad application in food computing tasks to date. Image to Text Models gradually play an important role in vision language tasks with the development of deep learning [20]. Models follow a main pipeline that encodes image input into intermediate stages and then decodes them into text output [49, 57]. Vinyals et al. [49] encoded the input image into a global visual vector through CNN and then applied RNN to generate captions, whose generalization on multi-task also supported the effect of the encoder-decoder pipeline. Xu et al. [57] further built an attention mechanism to pick the most related subregion vectors rather than depending only on the global visual vector. De-
spite the progress in the encoder-decoder pipelines, their ability is limited by only emphasis on single modularity input. Recent research switched to unimodality with the birth of large image-text datasets \cite{donahue2015long, radford2021learning}. CLIP \cite{radford2021learning} modified GPT-2 \cite{radford2019language} to obtain text features from textual input and used image-text contrastive learning, which trained the model with the similarity between the image and text. AL-BEF \cite{bepler2021efficient} utilized ViT (pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks) \cite{dosovitskiy2020image} as an image encoder and BERT \cite{devlin2018bert} as a text encoder to extract information with attention. One additional multimodal encoder was built on the extraction output, which additionally incorporated masked-language-modeling loss (MLM) to enhance the image-text interactions. Based on previous work, BLIP model \cite{liu2021blip}, designed to achieve unified vision-language understanding and generation, reached state-of-the-art results on various vision-language tasks in a zero-shot manner \cite{yang2021scaling}. By employing a captioner to generate synthetic captions and applying a filtering mechanism, BLIP maximizes the utilization of noisy web data. In our work, we leverage BLIP and ViT to generate food titles and ingredients separately.

**Text (Sequence) to Sequence Models** takes a sequence of text as input and maps it into a succession of another sequence \cite{vaswani2017attention}. For this task, previous work either built recurrent neural networks (RNNs) \cite{cho2014properties, hochreiter1997long, luong2015effective} or attention-based models \cite{devlin2018bert, radford2019language, liu2021blip}. GPT-2 \cite{radford2019language} is based on a transformer-decoder to perform tasks on various fields in a zero-shot setting, while T5 (text-to-text encoder-decoder model) \cite{raffel2020exploring} transformed text-related tasks in a text-to-sequence format to enhance its general ability. With the recent progress of large pre-trained LMs, prompting has become a popular and efficient approach to tackle many NLP tasks \cite{liu2020pre}. More specifically, few-shot prompting provided input-output mapping in demonstration to guide LMs to prompt the specific structure. Chain-of-thought (CoT) \cite{yu2020chain} with Self-Consistency \cite{maitre2021self} reached state-of-art on commonsense reasoning and symbolic reasoning, even compared to supervised models. Madaan et al. \cite{madaan2021chain} used graph demonstration to hint LMs generate complex Python classes for reasoning and state tracking. Capitalizing on these advancements, we employ T5 to generate cooking instructions from food ingredients and titles. To overcome resource and time constraints posed by large-scale food dataset, we also show how few-shot prompting can support food computing applications. Our work analyzes how to generate a curated recipe or convert recipe into a structured program flow \cite{zhao2022efficient} for further application with the help of prompting.

3. Proposed Methodology (FIRE)

FIRE consists of three components: (1) title generation from food images by using state-of-the-art image captioning, (2) ingredient extraction from images using vision transformers and decoder layers with attention, and (3) cooking instruction generation based on the generated title and extracted ingredients using an encoder-decoder model.

### 3.1. Title Generation

We generate recipe titles from food images using the BLIP model, a state-of-the-art image captioning approach. In our initial experiments with the off-the-shelf BLIP model, we observed promising results, yet, BLIP’s prediction accuracy was lower because of the domain shift between its training data and the food domain. Namely, BLIP tends to capture extraneous details impertinent to our goal because it was originally designed to provide a comprehensive image caption for a wide variety of settings. As an illustration, when presented with an image of a muffin, BLIP produced the description ‘a muffin positioned atop a wooden cutting board’. To better align the generated captions with recipe titles, we fine-tune the BLIP model using a subset of the Recipe1M dataset. We restrict our tuning to 10% of the training dataset, as fine-tuning on the entire dataset is computationally intensive \cite{li2020flexible}. We observe that the fine-tuned version of BLIP shows promising improvements in generating accurate, aligned, and pertinent titles for food images. The fine-tuned BLIP captions the same example image with a shorter string ‘muffin’, removing the additional extraneous information.

### 3.2. Ingredient Extraction

Extracting ingredients from a given food image presents challenges due to the inherent complexity and variability of
food compositions. Unlike generating titles or captions, determining a comprehensive and accurate list of ingredients requires a deep understanding of food characteristics, textures, and interactions. Additionally, the visual appearance of certain ingredients may overlap, whereas others may not be visible at all, leading to potential ambiguities and difficulties in discerning specific components. For instance, in a food image that contains a dish with melted cheese on top, from visual appearance alone it may be challenging to determine if the cheese used is mozzarella, cheddar, or any other type. These nuances motivate need for an architecture capable of retaining expressive embeddings from food images. While finetuned BLIP successfully generated accurate titles for food images, our analysis showed that using it for ingredient extraction led to significant hallucinations in the output. As BLIP is primarily trained for image captions, it struggles to generate ingredient lists accurately. To address this challenge, we develop an ingredient extraction pipeline (shown in Figure 2) built on top of the one proposed by [41].

**Feature Extractor:** We extract the image’s features by employing a vision transformer (ViT). ViT’s attention mechanism enables for effective handling of feature representations with stable and notably high resolution. This capability precisely meets the requirements of dense prediction tasks such as ingredient extraction from food images [63]. Furthermore, transformer-based approaches exhibit minimal reliance on the inductive bias, facilitating effective interaction and integration of long-range information. Unlike conventional CNNs, the output of a ViT is sequential; therefore, we use a fully connected (FC) linear layer to reshape the output and pass it to a 2D convolution (Conv2D) layer.

**Ingredient Decoder:** The feature extractor produces image embeddings. We pass these image embeddings through three normalization layers (layerNorm) and subsequently feed the output into our ingredient decoder responsible for extracting ingredients. The decoder consists of four consecutive blocks, each comprising multiple sequential layers: self-attention, conditional attention, two fully connected layers, and three normalization layers. In the last step, the decoder output is processed by a fully connected layer with a node count equivalent to the vocabulary size, resulting in a predicted set of embeddings.

Given a corpus with ingredients and recipes corresponding to food images, we construct a dictionary \( D \) consisting of \( N \) possible ingredients. Each recipe, \( r_i \), is associated with a set \( S \), comprising \( K \) ingredients selected from this dictionary. Given that the order of the ingredients does not affect the resulting recipe, we represent the ingredients as a set rather than a list. In other words, we exploit co-dependencies among ingredients without penalizing for prediction order. We represent the ingredient set \( S \) using a binary vector, \( s \), of dimension \( N \), where \( s_i = 1 \) if \( s_i \in S \); and 0 otherwise. Consequently, our training dataset consists of \( m \) pairs of image and ingredients sets: \( \{(x_i, s_i)\}_{i=0}^{m} \). In this case, the goal is to predict \( s \) from an image \( x \) by maximizing the following objective:

\[
\arg \max_{\theta_{img}, \theta_{ing}} \sum_{i=0}^{m} \log p(s_i = s_i|x_i; \theta_{img}, \theta_{ing})
\]

where \( \theta_{img} \) and \( \theta_{ing} \) represent the learnable parameters for the image encoder and ingredient decoder, respectively. While there may exist certain dependencies among the ingredients, such as the common combination of salt and pepper, these dependencies do not exert a dominant influence. Consequently, we can reasonably assume independence between the ingredients and factorize them as follows:

\[
\sum_{j=1}^{N} \log p(s_{ij} = s_{ij}|x_i) \leftrightarrow p(s_i = s_i|x_i)
\]

Our decoder makes ingredient predictions sequentially until it encounters an end-of-sequence (EOS) token. To mitigate the impact of the order of the ingredients, we aggregate the outputs separately across different time steps and use max pooling at the end to obtain the ingredient set. This enables training the model using binary cross-entropy loss \( \text{loss}_{\text{inpr}} \) between the predicted ingredients (after pooling) and the ground truth. However, since the EOS information is lost during pooling, we use a custom EOS loss \( \text{loss}_{\text{eos}} \). This loss calculates the binary cross-entropy between the predicted EOS probabilities at all time steps and the corresponding ground truth. Furthermore, to enhance performance, we incorporate a cardinality L1 penalty \( \text{loss}_{\text{card}} \), which constrains the length of the predicted ingredients to be close to the ground truth ingredients. We empirically find that integrating the \( \text{loss}_{\text{card}} \) leads to better performance.

\[
\text{loss} = \alpha_1 \times \text{loss}_{\text{inpr}} + \alpha_2 \times \text{loss}_{\text{eos}} + \alpha_3 \times \text{loss}_{\text{card}}
\]

where, \( \alpha_1=100 \), \( \alpha_2=1 \), and \( \alpha_3=1 \) are the hyper-parameters.

![Figure 3. Generating cooking instructions for a title and a set of ingredients.](image-url)

**3.3. Cooking Instruction Generation**

Considering the remarkable accomplishments of LMs in natural language applications like text generation and question answering [6], we pose cooking instruction generation as a language modeling task. Large LMs such as GPT [5], LLaMa [47], and Alpaca [38] are pre-trained with billions of tokens with multiple training objectives, which makes
them capable of understanding language in context. Refining the LMs for downstream tasks has demonstrated remarkable outcomes in various NLP assessments. While we expect that large LMs would be capable of generating cooking instructions after fine-tuning, they require prohibitive computational resources given their large number of parameters. Given the available resources and our research objective, we adopt popular encoder-decoder model, T5 [37], for generating cooking instructions. We conduct all experiments using base T5 model with 220M parameters. During finetuning, we pass title and ingredients of the recipe as a formatted string (see Figure 3), inspired by prior work [61].

The T5 is finetuned on three inputs: title, ingredients, and ingredients with quantity to incorporate maximum information from the dataset. However, we do not have ingredients with quantity at inference time; hence we can pass only the title and ingredients. Moreover, excluding the quantity information from our model ensures a fair comparison with previous approaches and investigates whether our model’s advantage stems from a well-structured architecture rather than relying solely on the augmentation of additional knowledge. By removing the influence of quantity information during inference, we aim to highlight the inherent capabilities of T5 and its ability to generate high-quality cooking instructions.

4. Experiment Setup

4.1. Dataset

Recipe1M is a large-scale dataset of over one million cooking recipes [26]. The dataset contains rich food-related information, including recipe titles, ingredients, cooking instructions, and nutritional information. The dataset itself is not provided publicly, instead, the authors provide a list of image URLs that can be scraped from the Web. While scraping this data, we encountered instances where we could not download images due to an expired URL or a corrupt image. Additionally, some of the recipes in the dataset did not have any accompanying images. Therefore, we only utilized recipes with at least one corresponding image available to us. Following the dataset filtering process, we obtained a training set comprising 259,932 samples, a development set containing 55,773 samples, and a test set consisting of 56,029 samples adequate for recipe analysis.

4.2. Baselines

Ingredient Extraction: We present a comparative analysis of FIRE’s ingredient extractor against two retrieval-based techniques: R_{2L} and R_{2L,R} [42]. R_{2L} learns joint embeddings between images and ingredient lists and uses them to retrieve the most relevant recipe within the embedding space. R_{2L,R} expands upon this approach by incorporating the joint embedding between recipe title, instructions, ingredients, and the corresponding food image to further enhance retrieval. We also compare our approach to two state-of-the-art generative models, namely FF_{TD} and InverseCooking [41]. FF_{TD} models the joint distribution of the ingredients set by utilizing the target distribution and greedily sampling from a cumulative distribution of sorted output probabilities until the sum of probabilities of selected elements exceeds a specified threshold. InverseCooking is an attention-based model that takes embeddings from ResNet50 as input and uses a transformer decoder architecture for ingredient generation.

Cooking Instruction Generation: As baselines, we utilize both InverseCooking [41] and Chef Transformer [29]. We specifically select these two baselines as their code was publicly available, and they work on recipe generation rather than retrieval from a database. InverseCooking is an end-to-end recipe generation model that takes food images as input and extracts ingredients, which along with image embeddings, are used to generate the title and the cooking instructions. Like our method, InverseCooking is also trained on Recipe1M. Chef Transformer is trained on the RecipeNLG [3] dataset and exclusively relies on ingredient inputs rather than food images for cooking instructions generation. Therefore, in the case of the Chef Transformer, we use ground truth ingredients for testing.

4.3. Evaluation Metrics

We evaluate FIRE on end-to-end cooking instruction generation and through ablation study on ingredient extraction. For end-to-end recipe generation, we employ document-level evaluation metrics: SacreBLEU and RougeL to assess the quality of our model’s output. Since the output of ingredient extraction models is a set, we evaluate their performance using F1-score/Dice score, and Jaccard/IoU similarity index, computed for accumulated counts of true positives, false negatives, and false positives over the entire dataset.

5. Results & Analysis

5.1. End-to-End Recipe Generation

The results in Table 1 show that FIRE exhibits superior performance compared to the two SotA baselines, InverseCooking and Chef Transformer. These results demonstrate our proposed pipeline’s ability to generate precise and coherent recipes, corroborating the effectiveness of FIRE and emphasizing the value of language generation models for high-quality recipe generation. These results also support our expectation that the FIRE method can generalize well without ingredient quantity information given at inference time, even when they were present during training. Meanwhile, we observe that training with extra information results in fewer hallucinations, especially regarding ingredi-
FIRE uses ground-truth ingredients for cooking instruction generation, informative signals to generate comprehensive recipes. Meanwhile, embeddings and automatically extracted ingredients during the cooking instruction generation phase. FIRE’s instruction generation language model relies on the recipe title and ingredients only, which provide FIRE with informative signals to generate comprehensive recipes.

As Chef Transformer does not support image input, it uses ground-truth ingredients for cooking instruction generation. In comparison, FIRE faces realistic challenges due to noisy ingredient extraction. Yet, FIRE easily outperforms Chef Transformer, and the gap increases further when FIRE is also provided with a ground-truth title and ingredient set. The low performance of Chef Transformer on this task can be attributed to its architecture and its reliance on just ingredients without any title information. As a set of ingredients can correspond to multiple recipes, the title may be crucial for disambiguation and coherence. For example, both Stir-fried Ginger Chicken and Garlic Ginger Chicken Soup share the same set of ingredients (chicken, garlic, soy sauce, and ginger). Despite this commonality, this same set of ingredients leads to two entirely different recipes.

### 5.2. Ablation Study

**Ingredient Extraction** The results on the ingredient extraction task are shown in Table 2. The retrieval-based approaches (R\textsubscript{12L} and R\textsubscript{12LR}) yield poor results. This can be expected, given their reliance on the presence of an exact matching recipe in the static dataset and their dependence on the dataset size and diversity. The models FF\textsubscript{T,D}, InverseCooking, and FIRE, which employ conditional generation, exhibit relatively higher performance in capturing ingredient information from food images. Moreover, out of these three models, FIRE achieves the highest IoU and F1 scores among all of the models, surpassing the second-ranked InverseCooking model with a relative margin of 1.5% in terms of IoU and 1.4% in terms of F1 score. We attribute this improvement to FIRE’s superior feature extraction capability that uses ViT rather than ResNet50.

**Image Feature Extraction** To understand the observed in-ingredient extraction gap between FIRE and InverseCooking, we compare the impact of image feature extractors on ingredient extraction. We ablate our feature extractor (ViT) with state-of-the-art CNN models: ResNet18, ResNet50, ResNet101, and InceptionV3. Table 3 reveals that ViT outperforms the other feature extractors, demonstrating its superior ability to capture and represent food image features relevant to ingredient extraction. Furthermore, to assess the feature extractor’s influence on the end-to-end FIRE pipeline, we substituted ViT with ResNet50. This change resulted in a performance decrease, as indicated in Table 1. This finding emphasizes the efficacy of leveraging the state-of-the-art feature extractor ViT for improved results in our food computing system.

**Zero-shot vs Fine-tuned** We compare the performance of zero-shot BLIP and T5 model against our fine-tuned model. The outcomes are detailed in Table 4. To assess the results for title generation by BLIP, we use a string similarity approach based on the longest common subsequence (LCS), as achieving an exact match is infrequent due to the vast array of recipe variations. For example, if the actual title is ‘black bean and rice salad’ and BLIP predicts ‘black bean and rice’, then a conventional accuracy metric would yield zero, whereas the LCS score would be 0.76. For T5, we utilize the SacreBLEU metric. The results demonstrate that SoTA BLIP and T5 models necessitate task-specific fine-tuning.

### Table 1. Recipe generation comparison on the test dataset. We report mean with one standard deviation of 10 experiments. **Bold** represents the best model. (*) represents the model tested on the ground truth title and ingredients to generate the recipe.

<table>
<thead>
<tr>
<th>Model</th>
<th>IoU</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chef Transformer [29]</td>
<td>4.61±0.32</td>
<td>17.54±0.19</td>
</tr>
<tr>
<td>InverseCooking [41]</td>
<td>5.48±0.21</td>
<td>19.47±0.15</td>
</tr>
<tr>
<td>FIRE (without loss\textsubscript{card})</td>
<td>5.91±0.17</td>
<td>20.87±0.13</td>
</tr>
<tr>
<td>FIRE (ResNet50)</td>
<td>5.87±0.10</td>
<td>20.49±0.08</td>
</tr>
<tr>
<td>FIRE</td>
<td>6.02±0.15</td>
<td>21.29±0.10</td>
</tr>
<tr>
<td>FIRE\textsuperscript{+}</td>
<td>7.29±0.11</td>
<td>25.17±0.07</td>
</tr>
</tbody>
</table>

### Table 2. Evaluation results on ingredient extraction using set metrics (IoU and F1). **Bold** represents the best model.

<table>
<thead>
<tr>
<th>Model</th>
<th>IoU</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>R\textsubscript{12L} [42]</td>
<td>18.92</td>
<td>31.83</td>
</tr>
<tr>
<td>R\textsubscript{12LR} [42]</td>
<td>19.85</td>
<td>33.13</td>
</tr>
<tr>
<td>FF\textsubscript{T,D} [41]</td>
<td>29.82</td>
<td>45.94</td>
</tr>
<tr>
<td>InverseCooking [41]</td>
<td>32.11</td>
<td>48.61</td>
</tr>
<tr>
<td>FIRE</td>
<td><strong>32.59</strong></td>
<td><strong>49.27</strong></td>
</tr>
</tbody>
</table>

### Table 3. Impact of SoTA feature extractors on ingredient extraction. All models are trained and tested on 10% dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>IoU</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet18</td>
<td>25.88</td>
<td>39.31</td>
</tr>
<tr>
<td>ResNet50</td>
<td>26.94</td>
<td>40.51</td>
</tr>
<tr>
<td>ResNet101</td>
<td>26.37</td>
<td>40.12</td>
</tr>
<tr>
<td>InceptionV3</td>
<td>25.31</td>
<td>38.92</td>
</tr>
<tr>
<td>ViT</td>
<td><strong>27.69</strong></td>
<td><strong>42.73</strong></td>
</tr>
</tbody>
</table>

### Table 4. Comparison between zero-shot (ZS) and fine-tuned (FT) versions of BLIP and T5.

<table>
<thead>
<tr>
<th>Model</th>
<th>ZS</th>
<th>FT</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLIP</td>
<td>17.89</td>
<td>37.72</td>
</tr>
<tr>
<td>T5</td>
<td>2.47</td>
<td>6.02</td>
</tr>
</tbody>
</table>
impact of adding this loss in model. Without cardinality loss, we believe the model struggles in realizing correct number of ingredients, which leads to divergence from the ground truth, thus lowering performance as shown in Table 1.

### 5.3. Error Analysis

In order to gain further insight into the performance of our recipe generation method, we inspected its performance on individual images. As shown in Figure 3, FIRE is often able to generate a correct recipe for dishes similar to those present in the Recipe1M dataset. Meanwhile, we also study its ability to provide a recipe for Pav bhaji, a popular Indian dish that is not present in the Recipe1M dataset. FIRE generates a recipe for "tomato and onion sandwich" as shown in Figure 4. As expected, the generated recipe is unrelated to the intended dish. Other state-of-the-art models are also not able to predict the correct recipe. We acknowledge the need for improvement in our model to better generalize to novel recipes. Meanwhile, we highlight the importance of developing better evaluation metrics. Conventional evaluation metrics such as SacreBLEU and ROUGE, failed to capture the accuracy of the generated recipes and detect certain text hallucinations. Given the significant impact of even a single mistake on the final outcome of a dish, it is crucial to develop a robust metric that can reliably ensure the completion of the desired cooking task beyond text similarity. For additional examples of both successful and unsuccessful cases, please refer to the Appendix.

### 6. FIRE Applications

While FIRE achieves state-of-the-art performance on the ambitious task of generating recipes from images, we go a step further and investigate its integration into larger pipelines for food computing applications. Namely, considering the promise of few-shot prompting of large language models, we describe how FIRE and large LMs can be integrated to support recipe customization and recipe-to-machine-code generation. For both applications, we provide a potential pipeline with an illustrated recipe and we conduct a pilot study to investigate its potential.

#### (1) Recipe Customization

Recipe customization is crucial due to the connection between food, customs, and individual preferences. Additionally, it becomes essential when addressing allergies or dietary restrictions. Surprisingly, despite the evident demand, existing literature lacks dedicated efforts in the domain of recipe customization. We are inspired by the Computer Cooking Contest (CCC) [32], an annual event showcasing computational systems that generate novel and creative recipes, enabling participants to employ AI and computational creativity in exploring innovative food combinations and techniques. However, we cannot use CCC directly because judges perform its evaluation manually. Our work aims to bridge the research gap by enabling personalized recipe customization, considering individual taste profiles and dietary restrictions. To guide future research in this area, we showcase the ability of FIRE to support a recipe customization approach that focuses on a wide range of topics (e.g., ingredient replacement, taste adjustment, calories adjustment, cooking time adaptation) to test few-shot performance thoroughly. As shown in the purple part of Figure 5, we perform ingredient removal to trim the potatoes from the recipe. Two sentences related to potatoes are deleted in the modified version, and one sentence is modified to ensure consistency. Specifically, we perform ingredient addition to replace ‘cheese’ with ‘cheddar cheese’ and recognize that it should be added before baking, resulting in the modified sentence ‘Sprinkle half each of cheddar cheese and onions.’ We manually design four demonstrations to hint GPT-3 to solve the customization requirements.

**Analysis** To assess the effectiveness of recipe customization, we conducted a human evaluation with seven experts involving 10 recipes and their customizations. Evaluators rated four attributes: efficacy, coherence, soundness, and proportions and measurements, on a 0 to 4 scale (0: strongly disagree, 1: disagree, 2: neutral, 3: agree, 4: strongly agree). Table 1 (Appendix) shows the average experimental results. On average, each attribute has a high result (3.5 to 3.76) with high Fleiss [10] kappa inter-annotator agreement (0.78 to 0.92). The results indicate the promise of integrating our method with few-shot LM prompting. Albeit provided with a limited number of demonstrations, the model can handle complex examples like *Can you make the food with fewer calories?* and replace milk with almond milk. We refer the reader to the Appendix for further details about the design, the data, and the results of the pilot experiment.

#### (2) Generating Machine Code for Image-based Recipes

Converting recipes to machine code enables automation, scalability, and integration with various systems, reducing manual intervention, resulting in savings in labor costs and reducing human errors while preparing the food. To fa-
cilitate this task, we combine FIRE’s recipe generation strength with the ability of large LMs to manipulate code-style prompts for structural tasks [25]. We show an example approach for generating Python-style code representations of recipes generated by FIRE, by prompting GPT-3 (orange part in Figure 5). This envisioned approach has two phases: code recipe prompting and symbolic triple refinement.

Code Recipe Prompting. We convert the output of FIRE into a Python-style prompt and leverage GPT-3 to generate code representations as shown in Figure 5. Further, we refine these representations into symbolic triple representations within a predefined space. For all input recipes \( r \in R \), we constructed corresponding Python-style prompting \( r^p \) and code representation \( c \). For any new recipe \( r' \), the input to the prompting pipeline was \( r^p_k \oplus c_k \oplus \cdot \cdot \cdot \oplus r^p_k \oplus c_k \oplus r' \), where \( k = 4 \) was the number of demonstrations and the output code representation \( c' \) is completion result of GPT-3.

Symbolic Triple Refinement. For further use in industrial applications, we refine code generation into symbolic triples \((i, r, o)\), where \( i \) and \( o \) represent the input list and output of operations, and \( r \) represented the cooking instruction and parameter details. This allows for a more structured and standardized representation of the generated code, facilitating easier integration with various applications.

Analysis. We conduct a similar human evaluation process focusing on how well ingredients, cooking instructions, and their descriptions are translated to code format on a scale of 0 (extremely poor) to 5 (excellent). Each property is rated on average between 4.27 and 4.47, with an inter-annotator agreement between 0.75 and 0.83. Despite the promising experiment results, few-shot prompting can produce hallucinations when tracking ingredients, especially in long contexts or when similar cooking tools are involved (e.g., saucepan, frying pan), which can be further explored by future work. We refer the reader to the Appendix for details about the design, the data, and the results of the pilot study.

7. Conclusion & Future Work

This paper introduced FIRE, a methodology tailored for food computing, focusing on generating food title, extracting ingredients, and generating cooking instructions solely from image inputs. We leveraged recent advancements in CV and language modeling to achieve superior performance against strong baselines. Furthermore, we demonstrated practical applications of FIRE for recipe customization and recipe-to-code generation, showcasing adaptability and automation potential of our approach. Experimental results validated the efficacy of FIRE, highlighting its promising prospects for future advancements and wide-ranging adoption in food computing. Inspired by our experiments, we list three challenges that should be addressed in future research:

1. A major limitation of both the proposed work and existing approaches lies in the absence of a reliable grounding mechanism [16] to ascertain the correctness of generated recipes. Conventional metrics are insufficient to capture this challenge. We propose to address this limitation by developing a metric that effectively captures the coherence and plausibility of generated recipes, providing a more comprehensive evaluation framework for recipe generation systems.

2. The diversity and availability of recipes are heavily dependent on the locations, climates, and religions [30, 44, 62], which prevent users from preparing food based on predefined recipes. One solution can be the injection of knowledge graphs [12, 43], which reflect the connection between the ingredients based on symbolic relations and contextual factors, thus informing the models about alternative ingredients.

3. Hallucination remains a critical challenge in recipe generation by natural language and vision models. We will investigate the possibility of incorporating methods for state tracking of participants [14,24] to enhance the production of reasonable and accurate results.
References


[25] Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. Language models of code are few-shot


[38] Markus Rokicki, Christoph Trattner, and Eleco Herder. The impact of recipe features, social cues and demographics on estimating the healthiness of online recipes. In *Proceedings of the international AAAI conference on web and social media*, number 1, 2018.


