Learning Program Representations for Food Images and Cooking Recipes

Dim P. Papadopoulos[1,3] Enrique Mora[2] Nadiia Chepurko[1] Kuan Wei Huang[1]
Ferda Ofli[4] Antonio Torralba[1]

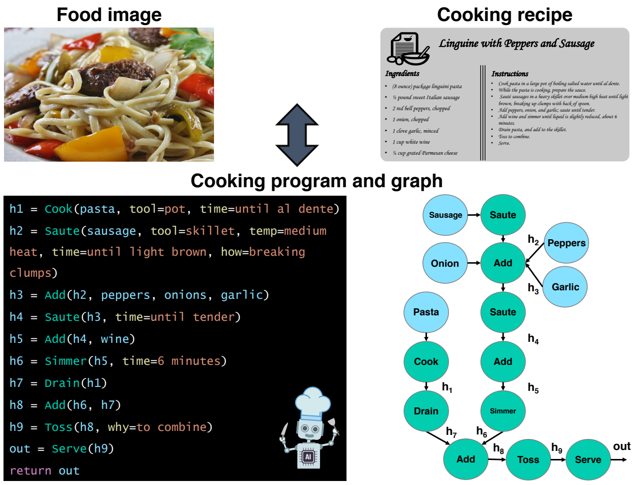
[1] MIT CSAIL [2] Nestle [3] DTU Compute [4] Qatar Computing Research Institute, HBKU
dimp@dtu.dk, enrique.mora@es.nestle.com, {nadiia,kwhuang,torralba}@mit.edu, fofli@hbku.edu.qa

Abstract

In this paper, we are interested in modeling a how-to instructional procedure, such as a cooking recipe, with a meaningful and rich high-level representation. Specifically, we propose to represent cooking recipes and food images as cooking programs. Programs provide a structured representation of the task, capturing cooking semantics and sequential relationships of actions in the form of a graph. This allows them to be easily manipulated by users and executed by agents. To this end, we build a model that is trained to learn a joint embedding between recipes and food images via self-supervision and jointly generate a program from this embedding as a sequence. To validate our idea, we crowdsource programs for cooking recipes and show that: (a) projecting the image-recipe embeddings into programs leads to better cross-modal retrieval results; (b) generating programs from images leads to better recognition results compared to predicting raw cooking instructions; and (c) we can generate food images by manipulating programs via optimizing the latent code of a GAN. Code, data, and models are available online[1].

1. Introduction

Food is an important part of our lives. Imagine an AI agent that can look at a dish and recognize ingredients and reliably reconstruct the exact recipe of the dish, or another agent that can read, interpret and execute a cooking recipe to produce our favorite meal. Computer vision community has long studied image-level food classification [4, 10, 21, 26, 27, 34], and only recently focused on understanding the mapping between recipes and images using multi-modal representations [14, 30, 48, 49, 67]. However, retrieval systems are limited to the existing database and usually fail for queries outside of this database while generating the full recipe from an image remains a challenge [47].



Linguine with Peppers and Sausage

Ingredients

- *(8 ounce) package linguini pasta*
- *½ pound sweet Italian sausage*
- *2 red bell peppers, chopped*
- *1 onion, chopped*
- *1 clove garlic, minced*
- *1 cup white wine*
- *¼ cup grated Parmesan cheese*

Instructions

- *Cook pasta in a large pot of boiling salted water until al dente.*
- *While the pasta is cooking, prepare the sauce.*
- *Sauté sausages in a heavy skillet over medium high heat until light brown, breaking up clumps with back of spoon.*
- *Add peppers, onion, and garlic; saute until tender.*
- *Add wine and simmer until liquid is slightly reduced, about 6 minutes.*
- *Drain pasta, and add to the skillet.*
- *Toss to combine.*
- *Serve.*

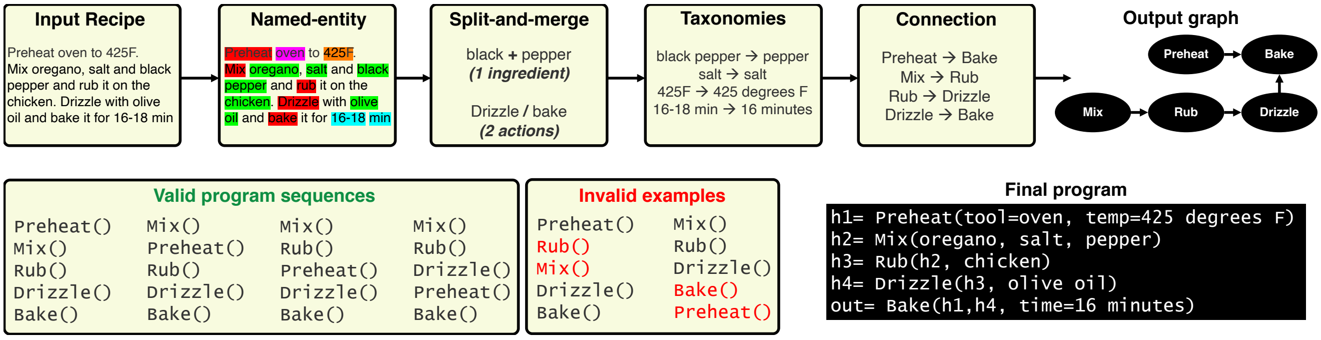
```
h1 = Cook(pasta, tool=pot, time=until al dente)
h2 = Saute(sausage, tool=skillet, temp=medium
heat, time=until light brown, how=breaking
clumps)
h3 = Add(h2, peppers, onions, garlic)
h4 = Saute(h3, time=until tender)
h5 = Add(h4, wine)
h6 = Simmer(h5, time=6 minutes)
h7 = Drain(h1)
h8 = Add(h6, h7)
h9 = Toss(h8, why=to combine)
out = Serve(h9)
return out
```

Figure 1. **Cooking programs.** We learn cooking programs from food images and recipes. Programs provide a structured representation of the cooking procedures which can also be represented as graphs (for brevity, we only show action and ingredient nodes).

Cooking recipes are step-by-step instructional procedures which we propose to represent as programs, capturing all the cooking semantics and relationships. A program contains a sequence of actions that can be written as functions (e.g., `Cook()`, `Add()`). Each action operates on specific ingredients under certain conditions, such as time or tool (e.g., `Cook(pasta, time='10 minutes', tool='pot')`). A program also captures the sequential dependency of the actions by maintaining their input-output connections. Note that even though cooking actions are often performed sequentially in time, their connections are not necessarily sequential. The program can also be represented as a graph where each function and parameter is a node, while the edges are the function connections or the connections between the parameters and the actions (Fig. 1).

Our goal is to generate cooking programs conditioned on food images or cooking recipes. We build a model that leverages the natural pairing of food images and recipes by

[1] http://cookingprograms.csail.mit.edu



```
Preheat()     Mix()         Mix()         Mix()
Mix()         Preheat()     Rub()         Rub()
Rub()         Rub()         Preheat()     Drizzle()
Drizzle()     Drizzle()     Drizzle()     Preheat()
Bake()        Bake()        Bake()        Bake()
```

```
Preheat()     Mix()
Rub()         Rub()
Mix()         Drizzle()
Drizzle()     Bake()
Bake()        Preheat()
```

```
h1= Preheat(tool=oven, temp=425 degrees F)
h2= Mix(oregano, salt, pepper)
h3= Rub(h2, chicken)
h4= Drizzle(h3, olive oil)
out= Bake(h1,h4, time=16 minutes)
```

Figure 2. **Annotation of cooking programs.** (Top) We obtain a graph from an input recipe via named-entity, split-and-merge parsing, taxonomy dictionaries and connection annotation. (Bottom) We obtain all valid program sequences from the graph and the final program.

learning a joint embedding using a vision and a text encoder. The visual and text embeddings are then used in a program decoder to generate cooking programs. Our model is trained end-to-end by jointly optimizing a ranking loss between the visual and text representations and two losses on the program sequence predictions. Because the sequence of some actions can be permuted without violating the input-output connections between the functions, we generate the set of all valid program sequences for each recipe (Fig. 2) and design a loss that operates on this set. At test time, the model can not only perform image-to-recipe retrieval tasks but can also predict the cooking program from an image or a recipe.

To validate our idea, we first crowdsource programs for cooking recipes selected from the Recipe1M dataset [49] using carefully designed tasks that can be easily performed by naive annotators. Experimental results show that our model leads to better cross-modal retrieval when it is jointly trained to generate programs. Moreover, generating programs leads to better food recognition results compared to predicting the raw cooking instructions. Finally, we show how to generate food images by manipulating cooking programs via optimizing the latent code of a GAN.

2. Related Work

Food recognition. Since the Food-101 dataset [4], food image analysis has become an established problem in computer vision [10, 12, 21, 26, 30, 34, 47, 48]. While early work focused mostly on image-level food categorization [4, 7, 27, 33], recent studies performed more fine-grained analyses such as ingredient recognition [7, 47], nutrition and calorie estimation [24, 32], food logging [31], and image generation [15, 38, 66]. Lately, cross-modal analysis of recipes and food images has become popular, thanks to the Recipe1M [49], where the authors tackled the image-to-recipe retrieval problem [30, 49]. Several studies improved the performance, including ACME [57], R^2-GAN [67], MCEN [14], SCAN [58], among others [5, 8, 13, 42, 48]. However, we aim to go beyond this task and understand the joint latent space of recipes and images so that we can generate new recipes from images and vice versa. Along these lines, [47] presented a two-step approach for predicting recipes from images: they first extract a list of ingredients given an image, and then, train a decoder that generates a recipe given both the image and the list of ingredients.

Recipe parsing. The task here is to parse a recipe and segment it into a sequence of individual actions. To achieve this, existing studies explored flow graphs [22, 35, 36, 61], tree-based solutions [6, 16], or extracting knowledge from cooking videos and transcripts without using the recipe text [60]. However, the datasets used in these studies are small (less than 300 recipes) and limited to only verbs and ingredients [6, 16]. More importantly, the parsing models require a specific and curated custom format for the recipes [16, 22]. Instead, we propose a program representation and describe how we can obtain programs for uncurated recipes [49] using efficient and scalable interfaces.

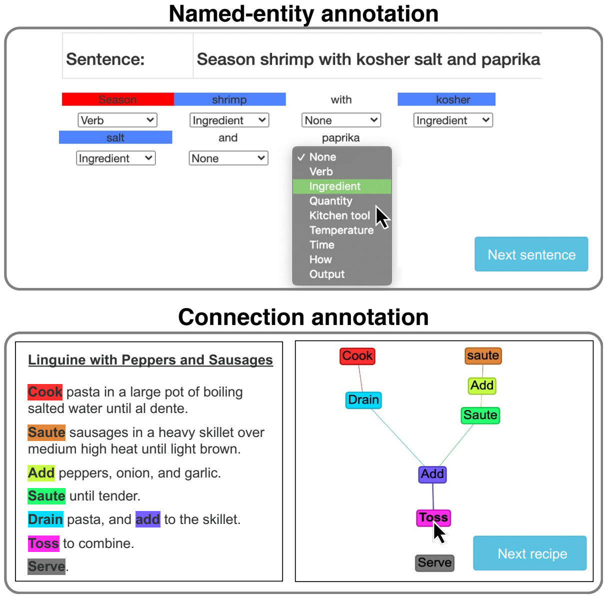
Program representation. Our work is also related to approaches that represent activities as programs and learn them from visual inputs (typically videos) [2, 37, 43, 62, 63]. This includes learning actions and objects by watching cooking videos [62, 63] or a sequence of atomic actions from instruction videos [2]. In [43], programs for household activities were used to train model that generate programs from videos or from natural language descriptions. Program generation methods have also been proposed for various tasks, such as visual question answering [17, 29, 64], fact verification [9], and geometry problem solving [28].

3. Cooking programs

In this section, we describe our program representation and show how we design a scalable crowdsourcing protocol to efficiently annotate recipes and obtain cooking programs.

3.1. Program scheme and graph representation

Cooking recipes consist of a title, a list of ingredients, and a list of instruction steps. Each step describes a cooking



Named-entity annotation

Sentence: Season shrimp with kosher salt and paprika

Season: Verb; shrimp: Ingredient; with: None; kosher: Ingredient; salt: Ingredient; and: None; paprika: ✓ None

Connection annotation

Linguine with Peppers and Sausages

Cook pasta in a large pot of boiling salted water until al dente.

Saute sausages in a heavy skillet over medium high heat until light brown.

Add peppers, onion, and garlic.

Saute until tender.

Drain pasta, and add to the skillet.

Toss to combine.

Serve.

Figure 3. **AMT interfaces. (Named-entity annotation)** We ask the annotators to tag every word of a cooking sentence. **(Connection annotation)** We ask the annotators to connect two actions (colored nodes) if there is an in-out relationship between them.

action that operates on ingredients using tools under certain conditions (e.g., time or temperature). This action results in an intermediate output that is used as an input in one of the following steps. This interesting structure, which resembles to the source code of a program, motivates us to use programs to represent cooking recipes (Fig. 1).

A program contains a sequence of functions that correspond to cooking actions. Each function takes as input a list of input variables (ingredients) and parameters (e.g., ingredient quantities or the way the action is performed, such as using a tool). For example, the sentence "Bake the chicken in the oven for 10 minutes at 400 degrees F" can be written as: `h = Bake(chicken, tool=oven, time=10 minutes, temp=400 degrees F);` The output of the function is denoted as h. For a full program, we also capture the input-output connections between the individual commands. Even though a recipe is performed sequentially in time, the output of an action is not always used as an input to the next action. Fig. 1 shows an example where the recipe consists of two sub-recipes that are combined at the end (i.e, cook pasta and prepare the sauce). These connections are captured by using the latent variables h as inputs to the corresponding functions (e.g. `Drain(h1)`).

Cooking graph. Cooking programs can be also represented as graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ (Fig. 1 (right)). Variables and functions are represented as vertices $\mathcal{V}$, while the edges $\mathcal{E}$ are the connections between the functions and their inputs. The graph shows that the sequence of actions can be permuted without violating input-output connections (i.e, the edges $\mathcal{E}$). We can compute all possible permutations that lead to valid, executable programs, i.e., h_i cannot be used as input before being computed. The bottom of Fig. 2 shows all valid permutations where the function `Preheat()` can be executed at any point before baking the chicken. In Sec. 4 we use this set of permutations to train our proposed model.

Program taxonomy. Parsing the recipes creates huge vocabularies for each category of the programs (Recipe1M contains 230k unique ingredients, 20k actions and 40k tools). To create a fixed and semantically meaningful vocabulary for each category, we follow a semi-automatic procedure. We describe here the process for the ingredients and we follow a similar one for the others: First, we use Sentence-BERT [45] and extract features for each unique ingredient. Then, we cluster them with K-Means using a high number of clusters (2,000) so that each cluster contains only semantically identical ingredients. Finally, for each cluster, we manually check its nearest neighbor and merge them if they are semantically almost identical. We repeat this iteratively until clusters cannot be merged anymore. Overall, this leads to 514 *ingredient*, 60 *action*, 55 *tool*, 130 *quantity*, 60 *temperature*, 152 *time*, 105 *how*, 112 *why* and 220 *output* clusters.

3.2. Program annotation

Crowdsourcing cooking programs is challenging as most naive annotators have no programming experience. For this reason, we split the process into four simple steps: (a) named-entity annotation, (b) split-and-merge parsing, (c) connection annotation and (d) program taxonomy (Fig. 2).

Named-entity annotation. In this step, we provide annotators a cooking sentence, and ask them to classify (tag) every word as one of the following categories: *cooking action*, *ingredient*, *quantity*, *kitchen tool*, *temperature*, *time*, *how*, *why*, and *output* (Fig. 3 (left)). The annotators first read instructions with the definition of each category and several examples. To ensure high quality responses, we design a protocol that follows common quality control mechanisms [25, 39, 46, 54, 56, 65] and monitor the performance of the annotators by using hidden pre-tagged sentences.

Split and merge. In this step, we perform two parsing operations without any human intervention. First, we split sentences that contain more than one action (e.g., "drizzle with olive oil and bake it for 16-18 minutes" in Fig. 2) so that every sentence contains only one action. Next, we merge tagged words into entities. For example, the phrase "mix fresh oregano, salt and black pepper" has five ingredient words but there are only three ingredients. Black pepper and fresh oregano are merged into the final variables.

Connection annotation. In this step, we provide annotators a recipe with highlighted actions and a panel with these actions as nodes (Fig. 3 (right)), and ask them to connect two

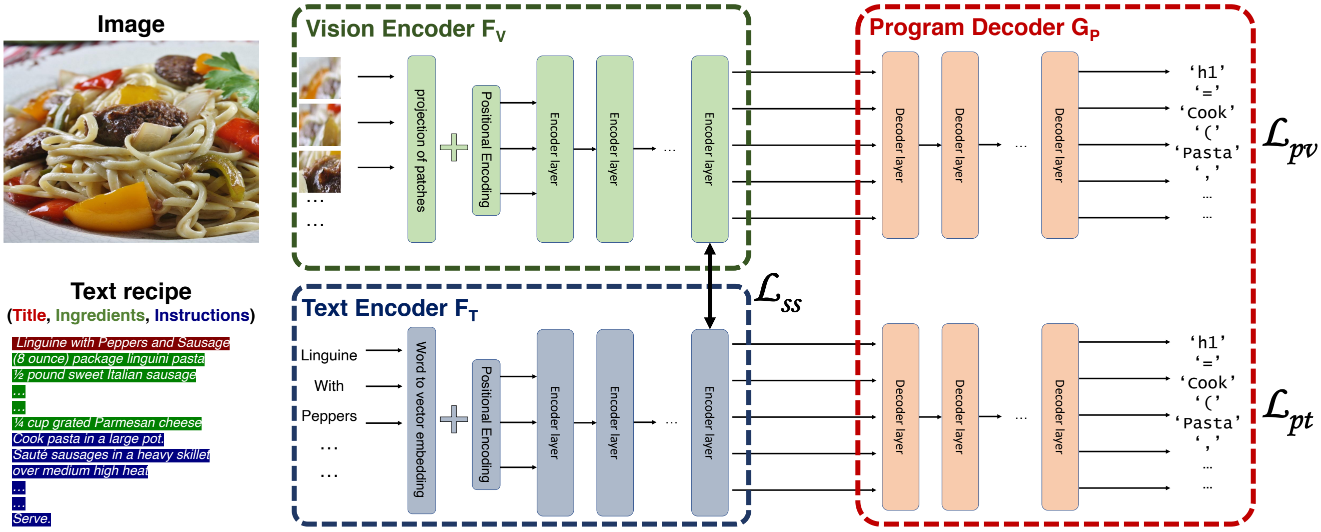


Figure 4. **Generating cooking programs from food images or cooking recipes.** Our model is trained end-to-end to project image and recipe features from a vision and a text encoder into a common space and to jointly generate programs as a sequence of commands using a program decoder conditioned on the image and text features.

nodes with an edge if there is an input-output relationship between them. To ensure high quality, we provide instructions and follow standard quality control mechanisms.

Program taxonomies. In this step, we follow the variable taxonomies (Sec. 3.1) and for each category (i.e, actions, ingredients, tools, etc.) we map all obtained values of each node into the corresponding categories from our fixed program vocabulary (e.g., "black pepper"→"pepper" in Fig. 2).

Final program. Finally, we follow the rules described in Sec. 3.1 to create the final program and the corresponding graph. The output of the last action is denoted as `out`.

3.3. Program collection

We ran experiments on Amazon Mechanical Turk (AMT) and collected programs for 3,708 recipes selected from the Recipe1M dataset [49][2]. This translates to 42,473 sentences with 478,285 tagged words and 54,154 annotated edge connections. An annotated example is shown in Fig. 2.

Annotation time and quality. The median response time of the named-entity task was 17 s per sentence, while the time of the connection task was 75 s per recipe. The annotated recipes have on average 11 sentences leading to a total annotation time of 4 minutes per recipe. The total cost for annotating our programs was about $2,000 (hourly wage about $8). To analyze and quantify the quality of our annotations, we annotate 50 recipes (550 sentences) once more using different annotators and measure the human agreement. The human agreement of the named-entity task is 97.9% (i.e., words with the same annotation label) . For the connection task, we found that 7.3% of the connections do not appear in both annotations (agreement of 92.7%).

4. From food images and recipes to programs

We introduce the novel task to generate cooking programs from food images or cooking recipes. In Sec. 4.1, we present the model architecture, while in Sec 4.2, we explain the training process and the objective functions.

4.1. Model architecture

Given a set of images $\mathcal{I}$ paired with their recipes $\mathcal{R}$ and a set of programs P, our goal is to learn how to generate a cooking program conditioned on an image or a recipe. Our model consists of three components: (a) a vision encoder, (b) a text encoder, and (c) a program decoder (Fig. 4). The model is trained to embed the images and recipes into the same space via self-supervision and to generate a program from an image or a recipe embedding.

Vision encoder. Food images are fed into the vision encoder F_V based on the Vision Transformer (ViT) [11]. The image is split into fixed-size patches (tokens). After a linear projection and adding position embedding, these tokens are fed to a stack of k_v Transformer encoder layers [55]. Unlike the original ViT where the image representation is obtained from the "classification token", we obtain it from the features of all patches after average pooling.

Text encoder. The text from cooking recipes is fed into the text encoder F_T [55]. The architecture of F_T is similar to F_V with the difference that here the words play the role of the tokens fed into k_t identical Transformer layers [55]. Once more, the final recipe representation is obtained after an average pooling layer on top of the token embeddings.

[2] Human subject experiments were conducted with an IRB approval.

Recipe Component	Program Loss	Encoder Layers	image-to-recipe medR	 R@1	 R@5	 R@10	recipe-to-image medR	 R@1	 R@5	 R@10
Title		8 (small)	4.6	24.0	54.4	67.8	4.5	23.8	54.2	67.5
Ingredients		8 (small)	2.8	39.1	68.1	80.3	3.0	38.9	68.4	80.5
Instructions		8 (small)	3.1	36.6	65.2	76.8	3.0	36.5	65.8	76.9
Title+Ingr		8 (small)	2.0	43.6	75.6	85.0	2.0	44.9	76.2	85.4
Title+Ingr+Inst		8 (small)	1.0	53.5	81.8	89.2	1.0	53.1	82.0	89.6
Title+Ingr+Inst	✓	8 (small)	1.0	58.6	85.7	91.7	1.0	58.2	85.5	92.0
Title+Ingr+Inst	✓	12 (base)	**1.0**	**66.9**	**90.9**	**95.1**	**1.0**	**66.8**	**89.8**	**94.6**

State-of-the-art cross-modal retrieval	Img Encoder	image-to-recipe medR	 R@1	 R@5	 R@10	recipe-to-image medR	 R@1	 R@5	 R@10
Salvador CVPR 17 [49]	ResNet-50	5.2	24.0	51.0	65.0	5.1	25.0	52.0	65.0
Chen ACM MM 18 [8]	ResNet-50	4.6	25.6	53.7	66.9	4.6	25.7	53.9	67.1
Carvalho SIGIR 18 [5]	ResNet-50	2.0	39.8	69.0	77.4	1.0	40.2	68.1	78.7
Zhu CVPR 19 [67]	ResNet-50	2.0	39.1	71.0	81.7	2.0	40.6	72.6	83.3
Fu CVPR 20 [14]	ResNet-50	2.0	48.2	75.8	83.6	1.9	48.4	76.1	83.7
Pham AAAI 21 [42]	ResNet-50	1.6	49.7	79.3	86.3	1.6	50.1	79.0	86.4
Wang CVPR 19 [57]	ResNet-50	1.0	51.8	80.2	87.5	1.0	52.8	80.2	87.6
Wang TMM 21 [58]	ResNet-50	1.0	54.0	81.7	88.8	1.0	54.9	81.9	89.0
Fain arXiv 19 [13]	ResNeXt-101	1.0	60.2	84.0	89.7	–	–	–	–
Salvador CVPR 21 [48]	ResNet-50	1.0	60.0	87.6	92.9	1.0	60.3	87.6	93.2
Salvador CVPR 21 [48]	ViT-B/16	1.0	63.2	88.3	93.1	–	–	–	–
Ours	**ViT-B/16**	**1.0**	**66.9**	**90.9**	**95.1**	**1.0**	**66.8**	**89.8**	**94.6**

Table 1. **Cross-modal retrieval results on Recipe1M.** At the top part of the table, we report ablation studies of our model, while at the bottom part, we compare our results with the state-of-the-art cross-modal retrieval approaches.

Program decoder. The program decoder G_P consists of k_p transformer decoder layers [55]. The features obtained from F_V are fed into the multi-head attention of each layer following the standard attention mechanism [55]. This results in a predicted program sequence P_v given the image. We repeat the approach for the text part. As such, the features obtained from F_T are fed to G_P to predict a program sequence P_t. During inference, a program can be predicted given either a food image or a recipe using the corresponding features and the same decoder G_P (shared weights).

Loss hyperparameters	ViT features	image-to-recipe R@1	 R@5	 R@10
$\lambda_{pv} = 0.1, \lambda_{pt} = 0.1$	average	**58.6**	**85.7**	**91.7**
$\lambda_{pv} = 0.1, \lambda_{pt} = 0.1$	cls token	56.1	83.8	90.1
$\lambda_{pv} = 0, \lambda_{pt} = 0$	average	53.5	81.8	89.2
$\lambda_{pv} = 1, \lambda_{pt} = 1$	average	58.0	85.1	90.9
$\lambda_{pv} = 10, \lambda_{pt} = 10$	average	56.4	83.7	89.6

Table 2. **Ablation study** on ViT feature representation and loss hyperparameters. All models use encoders with 8 layers (small).

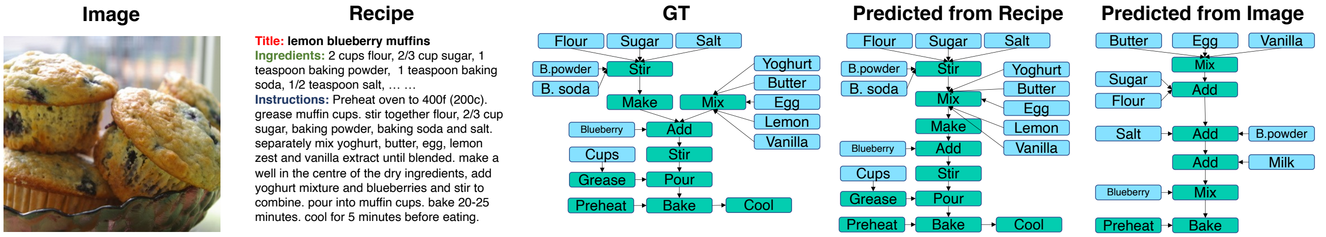
4.2. Training model and loss functions

We train our model end-to-end to jointly learn to project the image features from F_V and the text features from F_T into a common space through a self-supervised loss $\mathcal{L}_{ss}$ and to generate programs matching the ground-truth ones from the food images $\mathcal{L}_{pv}$ and the cooking recipes $\mathcal{L}_{pt}$.

Self-supervised triplet loss. We follow prior work and use a bi-directional max-margin triplet ranking loss to project images and text descriptions into a joint space [18,19,53,59] due to its recent success at the image-to-recipe retrieval task [5, 48, 57, 67]. The loss is based on the triplet ranking loss [51, 52] $\mathcal{L}_t(a, p, n) = max(0, s(f(a), f(n)) - s(f(a), f(p)) + m)$ where a is an anchor input, p and n are positive and negative samples, s is a similarity function, f is an embedding and m is a fixed constant margin.

For a mini-batch with size N of image-recipe pairs $\{I_i, R_i\}_{i=1}^{N}$, we obtain the feature representations of the image I_i and a recipe R_j from the visual F_V and text encoder F_T as $F_V(I_i)$ and $F_T(R_j)$, respectively. The image-recipe pair is considered as positive when $i = j$ and as negative otherwise. The final self-supervised bi-directional loss of the mini-batch, which computes $\mathcal{L}_t$ twice, is given by:

$$\begin{aligned}\mathcal{L}_{ss} = \frac{1}{N} \sum_{i=1, j\neq i}^{N} & max(0, s(i,j) - s(i,i) + m) + \\ & + max(0, s(j,i) - s(i,i) + m) \qquad (1)\end{aligned}$$



Title: **lemon blueberry muffins**
Ingredients: 2 cups flour, 2/3 cup sugar, 1 teaspoon baking powder, 1 teaspoon baking soda, 1/2 teaspoon salt, … …
Instructions: Preheat oven to 400f (200c). grease muffin cups. stir together flour, 2/3 cup sugar, baking powder, baking soda and salt. separately mix yoghurt, butter, egg, lemon zest and vanilla extract until blended. make a well in the centre of the dry ingredients, add yoghurt mixture and blueberries and stir to combine. pour into muffin cups. bake 20-25 minutes. cool for 5 minutes before eating.

Figure 5. **Generating programs from food images and recipes.** We show the output graphs of the two generated programs conditioned on the input recipe or the input image. For visualization purposes, we only show action (green) and ingredient (cyan) nodes.

where $s(i,j)$=$s(F_V(I_i), F_T(R_j))$ denotes the cosine similarity between image I_i and a recipe R_j.

Program prediction loss. Each recipe R_i is a sequence of K sentences $(r^1, r^2, ..., r^K)$. The program P_i is a sequence of L programming function commands $(p^1, p^2, ..., p^L)$ following the ordering of the recipe instructions (Fig. 1). However, as described in Sec. 3.1, some of the commands p^k can be permuted without violating the input-output connections between the functions. We denote as $\mathcal{P}_i = (P_i^0, P_i^1, ...P_i^\beta)$ the set of all β valid permutations of the sequence P_i. At each time step, G_p predicts a probability distribution for the output tokens, which is typically followed by a softmax and a cross-entropy loss between the predicted and the target sequence. Here, we expand the cross-entropy loss to handle a set of multiple candidate sequences $\mathcal{P}$.

Let $\Pi_i^I = G_p(F_V(I_i))$ be the generated program condition on I_i and $\Pi_i^R = G_p(F_T(R_i))$ be the generated program condition on R_i. The loss $\mathcal{L}_{pv}$ for a mini-batch is given by:

$$\mathcal{L}_{pv} = \frac{1}{N}\sum_{i=1}^{N} \min_{j\in[1,\beta]} \mathcal{L}_{ce}(\Pi_i^I, P_i^j) \qquad (2)$$

where $\mathcal{L}_{ce}(\Pi_i^I, P_i^j)$ is the cross-entropy loss between Π_i and the j^{th} target program sequence P_i^j. $\mathcal{L}_{pt}$ is given by eq. (2) by replacing Π_i^I with Π_i^R.

Final loss. The full objective function $\mathcal{L}$ of our model is defined as $\mathcal{L}$ = $\lambda_{ss}\mathcal{L}_{ss}$ + $\lambda_{pv}\mathcal{L}_{pv}$ + $\lambda_{pt}\mathcal{L}_{pt}$, where λ_{ss}, λ_{pv}, λ_{pt} are hyperparameters that control the relative importance of the image-recipe loss to the program prediction losses.

5. Experimental results

This section presents our experimental results. We evaluate our approach on three tasks: image-to-recipe retrieval (Sec. 5.1), program generation from recipes and images (Sec. 5.2), and image generation from programs (Sec. 5.3).

Data. We use the Recipe1M[3] dataset [49] as standard in previous work [13,14,48,49,57,58,67]. Recipe1M contains 887,706 food images and 1,029,720 cooking recipes split in training (70%), validation (15%) and test (15%) sets. Note that we use only the recipes that have corresponding images (340,831 recipes). For the cooking programs, we use our ground-truth dataset with 3,708 programs.

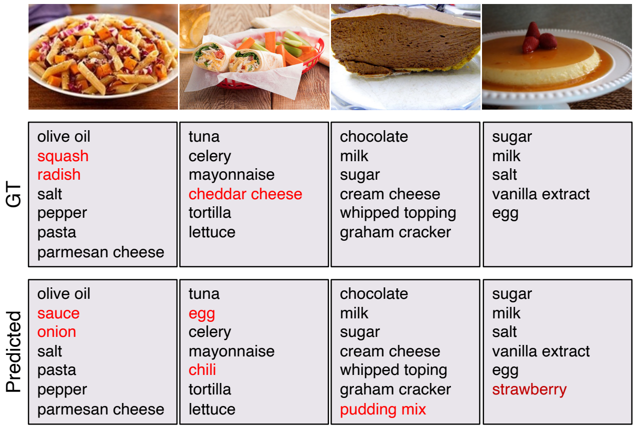
Implementation details. Unless stated otherwise, we use the following settings. F_V is based on ViT-B/16 [11] (k_v=12 layers, 12 heads) originally pretrained on ImageNet [46]. F_T and G_P are based on [55]. We use k_t=8 encoder layers (8 heads) and 2 decoder layers (4 heads). To handle the imbalance between the image-recipe and the recipe-program pairs, we first pre-train F_T and G_P alone using our annotated recipe-program pairs and obtain pseudo ground-truth programs for the whole Recipe1M. The images are resized to 256×256 and then cropped to 224×224. During training we perform random crop and horizontal flip augmentation. We train our model for 50 epochs using the Adam optimizer [23] with a base learning rate of 10^{-4} and

Input: Food images				
	Ingredients (F1 ↑)	**Actions (F1 ↑)**	**Tools (F1 ↑)**	**Full graph* (GED ↓)**
Random image	12.6	14.6	14.2	102.1
Retrieved image	39.4	51.6	66.9	79.1
NN (oracle)	53.5	66.5	81.1	62.2
Instructions	28.5	38.3	50.5	–
Programs (CE)	52.8	64.5	78.1	72.1
Programs (minCE)	53.5	64.7	78.1	67.2

Input: Cooking recipes				
	Ingredients (F1 ↑)	**Actions (F1 ↑)**	**Tools (F1 ↑)**	**Full graph* (GED ↓)**
Random recipe	12.4	14.5	14.2	101.5
Retrieved recipe	43.4	55.2	74.2	67.1
NN (oracle)	53.5	66.5	81.1	57.2
Instructions	41.6	49.3	66.6	–
Programs (CE)	75.4	83.1	83.8	19.1
Programs (minCE)	75.5	83.1	84.1	16.8

Table 3. **Evaluation of the predicted programs from images (top) and from recipes (bottom)**. For the full program, we report the graph edit distance (GED) between the ground-truth and the predicted graphs. We also extract the ingredients, actions and tools from the programs and report the F1 score with respect to the ground truth. *Note that due to the high computational cost of graph matching, GED is computed only on 5% of the test set.

[3] obtained from http://im2recipe.csail.mit.edu/



GT	olive oil squash radish salt pepper pasta parmesan cheese	tuna celery mayonnaise cheddar cheese tortilla lettuce	chocolate milk sugar cream cheese whipped topping graham cracker	sugar milk salt vanilla extract egg
Predicted	olive oil sauce onion salt pasta pepper parmesan cheese	tuna egg celery mayonnaise chili tortilla lettuce	chocolate milk sugar cream cheese whipped toping graham cracker pudding mix	sugar milk salt vanilla extract egg strawberry

Figure 6. **Ingredient prediction from images.** We show qualitative results for the ingredients extracted from generated programs (P_v). The ingredients highlighted in red are either FP or FN.

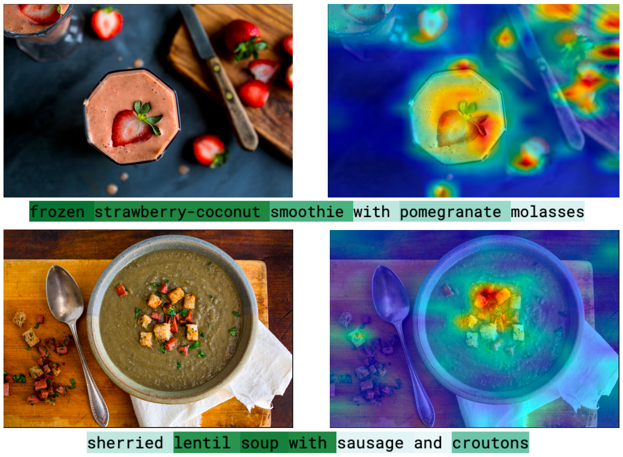

frozen strawberry-coconut smoothie with pomegranate molasses
sherried lentil soup with sausage and croutons

Figure 7. **Attention.** We use Attention Rollout [1] to visualize the attention weights on images and on recipe titles.

a step decay of 0.1 every 20 epochs. We set $\lambda_{ss} = 1$, $\lambda_{pv} = 0.1$, $\lambda_{pt} = 0.1$ and $m = 0.3$. All experiments were run on four Nvidia Titan X GPUs.

5.1. Image-to-recipe retrieval

Evaluation. Following the protocol of Recipe1M [49], we measure the retrieval performance on the test set with median rank (medR) and recall at top K (R@K) for $K = 1, 5, 10$ on ranking of 1,000 recipe-image pairs. We report the average metrics after repeating experiments 10 times.

Recipe components. We first examine the effect of training and testing the model using different recipe components. In the first three rows of Tab. 1, we observe that the ingredient list is the most informative component (R@1=39.1). Combining it with the title gives a small boost in performance (+4.5 in R@1), while using the full recipe yields 53.5.

Predicting programs. The first five models were trained without the decoder G_P. We observe that training the model using G_P to predict programs from the common embedding space significantly improves the performance (+4.9 R@1).

Deeper encoders. When using deeper encoders (*Base* model with 12 layers), the R@1 accuracy jumps to 66.9.

Comparison with state of the art. In Tab. 1(bottom), we compare our final model against several approaches. We achieve state-of-the-art performance thanks to the powerful transformer encoders and our program prediction loss. Note that a fair comparison among all methods is hard since they all use different image (ResNet, ResNext, ViT) and text (LSTMs, transformers) encoders. Also, note that previous work [48] use multiple transformer encoders for each component of the recipe. Instead, we use only a single text encoder (Fig. 4) leading to a much simpler and efficient model. Also, we do not use any unpaired data from Recipe1M unlike most of the existing approaches [5, 13, 48, 49, 57, 67].

ViT features. Using features from the "cls token" as in the original ViT [11] (instead of average pooling) causes a drop in performance (-2.5 in R@1), as shown in Tab. 2(top).

Loss hyperparameters. In Tab. 2, we perform an ablation study for λ_{pv} and λ_{pt} showing a small performance drop (-0.6% with $\lambda_{pv} = 1$ and -2.2% with $\lambda_{pv} = 10$ in R@1).

5.2. Program prediction from images and recipes

We evaluate here our model on the task of predicting programs from images or recipes (Fig. 5, Tab. 3).

Evaluation. Instead of considering the program as a sequence and use standard sequence evaluation metrics [40], we turn the programs into graphs and measure the graph edit distance (GED) [50] wrt to the groundtruth one. GED captures not only the semantic nodes but also their ordering and topology. This evaluation is feasible due to our fixed program vocabulary. Moreover, we extract only the ingredient nodes of the graphs and measure accuracy with the F1 score between the predicted and the ground-truth sets. We follow the same strategy for the action and tool nodes.

Programs from images. In Tab. 3(top), we report our results for predicting programs from images and compare with several baselines: (a) *Random image*: program from a random image; (b) *Retrieved image*: program from the top retrieved image using our retrieval system; (c) *NN (oracle)*: oracle nearest neighbor (NN) program (e) *Instructions*: our model where we decode the instruction text instead of a program. Tab. 3(top) shows that generating programs leads to better results (better ingredient, action and tool prediction) than the strong baselines of decoding sentences or relying on a retrieval system for making predictions. Interestingly, our model is only slightly below the oracle baseline that simulates an ideal retrieval system. We also observe that our program loss (minCE) slightly improves results over the standard CE loss. Fig. 6 show qualitative results for the ingredient prediction on food images, while Fig. 5 shows a

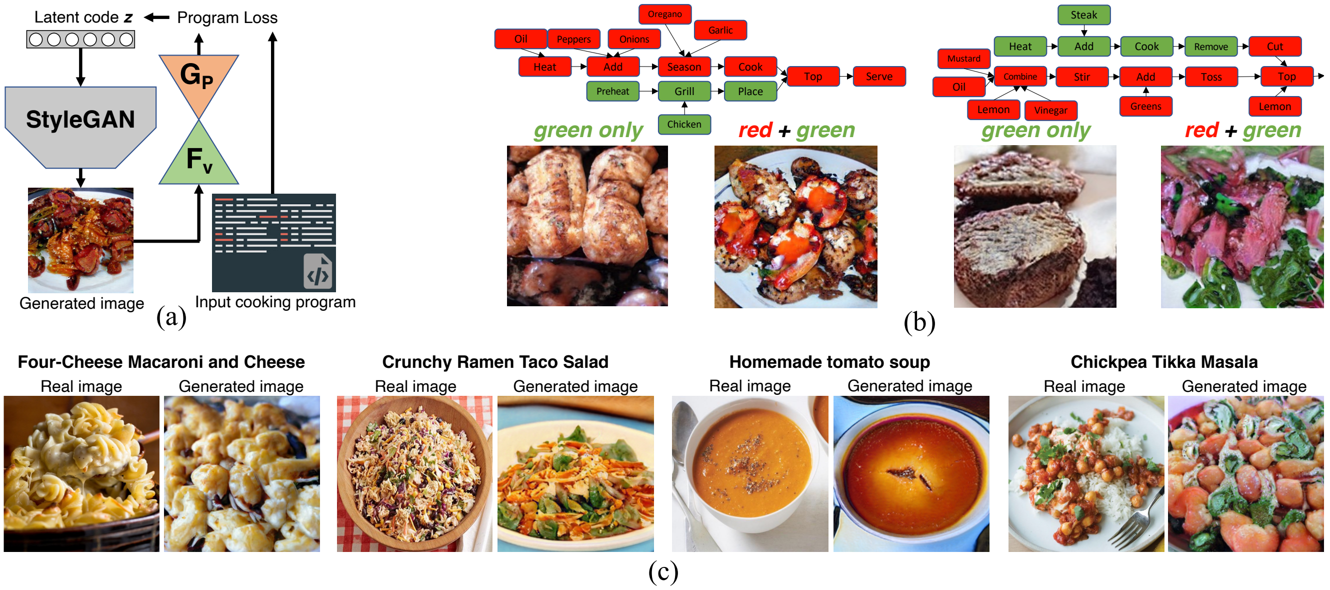


Figure 8. **Generating food images from programs.** (a) We optimize the latent code of a GAN by computing a loss between the input program and the predicted one from a generated image. (b) Generating images by manipulating programs. The images on the left of the graphs are generated using only the green nodes, while images on the right are obtained from the full graph. (c) Examples of generated images from ground-truth programs and comparison with the real images.

predicted program from a recipe and an image.

Programs from recipes. Similarly, in Tab. 3(bottom), we report our results for predicting programs from recipes and repeat the above baselines. The *text classifier* here is a transformer-based multi-label classifier that predicts ingredients. As expected, predicting programs from recipes is a much simpler task than predicting them from images. Once more, we observe similar trends and our loss yields slightly better results than the standard CE and significantly better than the *Instructions* baseline (+33.9% F1 for ingredients).

Attention. To further understand how F_V and F_T process images and text, we visualize their attention weights following [1]. In Fig. 7 we observe that both encoders focus on semantically similar concepts across the two domains.

5.3. Image generation

We show an interesting application of image generation conditioned on cooking programs using our model. We first train a StyleGANv2 [20] on the images of the Recipe1M dataset [49]. Given an initial latent vector z_0, we use the GAN to generate an image. The image goes through the encoder F_V and the decoder G_P to obtain a program. We compute the loss between this program and the desired input one. We use the loss to optimize z via backpropagation. Similar approaches have been proposed using the recently introduced CLIP [44] model to drive image generation using text sentences [3, 41]. Fig. 8(left) illustrates our approach while in Fig. 8(right) we show examples of generated images using a full program or a part of it (green nodes only). We observe that the generated images are realistic and capture plausibly the content of the input program.

6. Conclusions

We proposed to model cooking recipes with cooking programs. We designed a program scheme and annotated a set of programs for cooking recipes and we presented an approach for learning to predict programs from food images and recipes. Experimental results showed that projecting the common space between images and recipes into programs can improve retrieval results. Finally, we showed how we can generate food images by manipulating a program. We hope that programs will open new directions such as allowing agents to execute recipes or allowing us to extract common-sense knowledge for food from graphs.

Limitations and societal impact. In this work, we do not go beyond predicting the cooking procedure via programs. However, predicting nutritional value, estimating calories and their impact in our health is an important topic. Moreover, inaccurate predicted programs might not be able to be executed or lead to inedible food. Also, ingredient prediction models should be applied consciously especially in user cases with food allergies. Future work involves an analysis on the potential biases that the programs or our training data might have (e.g. towards unhealthy food dishes, towards western world with underrepresented cuisines) and the impact that this might have on the food industry.

Acknowledgments. This work is supported by Nestlé.

References

[1] Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In *ACL*, 2020. 7, 8

[2] Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Josef Sivic, Ivan Laptev, and Simon Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *CVPR*, pages 4575–4583, 2016. 2

[3] David Bau, Alex Andonian, Audrey Cui, YeonHwan Park, Ali Jahanian, Aude Oliva, and Antonio Torralba. Paint by word. *arXiv preprint arXiv:2103.10951*, 2021. 8

[4] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *ECCV*, 2014. 1, 2

[5] Micael Carvalho, Rémi Cadène, David Picard, Laure Soulier, Nicolas Thome, and Matthieu Cord. Cross-modal retrieval in the cooking context: Learning semantic text-image embeddings. In *SIGIR*, 2018. 2, 5, 7

[6] Minsuk Chang, Léonore V Guillain, Hyeungshik Jung, Vivian M Hare, Juho Kim, and Maneesh Agrawala. Recipescape: An interactive tool for analyzing cooking instructions at scale. In *CHI*, pages 1–12, 2018. 2

[7] Jingjing Chen and Chong-Wah Ngo. Deep-based ingredient recognition for cooking recipe retrieval. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 32–41, 2016. 2

[8] Jing-Jing Chen, Chong-Wah Ngo, Fu-Li Feng, and Tat-Seng Chua. Deep understanding of cooking procedure for cross-modal recipe retrieval. In *Proceedings of the 26th ACM international conference on Multimedia*, 2018. 2, 5

[9] Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. TabFact: A large-scale dataset for table-based fact verification. In *ICLR*, 2020. 2

[10] Xin Chen, Yu Zhu, Hua Zhou, Liang Diao, and Dongyan Wang. Chinesefoodnet: A large-scale image dataset for chinese food recognition. *arXiv preprint arXiv:1705.02743*, 2017. 1, 2

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 4, 6, 7

[12] Martin Engilberge, Louis Chevallier, Patrick Pérez, and Matthieu Cord. Finding beans in burgers: Deep semantic-visual embedding with localization. In *CVPR*, pages 3984–3993, 2018. 2

[13] Mikhail Fain, Andrey Ponikar, Ryan Fox, and Danushka Bollegala. Dividing and conquering cross-modal recipe retrieval: from nearest neighbours baselines to sota. *arXiv preprint arXiv:1911.12763*, 2019. 2, 5, 6, 7

[14] Han Fu, Rui Wu, Chenghao Liu, and Jianling Sun. Mcen: Bridging cross-modal gap between cooking recipes and dish images with latent variable model. In *CVPR*, 2020. 1, 2, 5, 6

[15] Fangda Han, Ricardo Guerrero, and Vladimir Pavlovic. Cookgan: Meal image synthesis from ingredients. In *WACV*, 2020. 2

[16] Jermsak Jermsurawong and Nizar Habash. Predicting the structure of cooking recipes. In *EMNLP*, pages 781–786, 2015. 2

[17] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *ICCV*, 2017. 2

[18] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015. 5

[19] Andrej Karpathy, Armand Joulin, and Li Fei-Fei. Deep fragment embeddings for bidirectional image sentence mapping. In *NeurIPS*, 2014. 5

[20] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020. 8

[21] Parneet Kaur, Karan Sikka, Weijun Wang, Serge Belongie, and Ajay Divakaran. Foodx-251: a dataset for fine-grained food classification. *arXiv preprint arXiv:1907.06167*, 2019. 1, 2

[22] Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. Mise en place: Unsupervised interpretation of instructional recipes. In *EMNLP*, pages 982–992, 2015. 2

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[24] Mandy Korpusik and James Glass. Spoken language understanding for a nutrition dialogue system. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2017. 2

[25] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018. 3

[26] Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. In *CVPR*, pages 5447–5456, 2018. 1, 2

[27] Chang Liu, Yu Cao, Yan Luo, Guanling Chen, Vinod Vokkarane, and Yunsheng Ma. Deepfood: Deep learning-based food image recognition for computer-aided dietary assessment. In *International Conference on Smart Homes and Health Telematics*, 2016. 1, 2

[28] Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. Inter-GPS: Interpretable Geometry Problem Solving with Formal Language and Symbolic Reasoning. In *ACL*, 2021. 2

[29] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *ICLR*, 2019. 2

[30] Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):187–203, 2019. 1, 2

[31] Michele Merler, Hui Wu, Rosario Uceda-Sosa, Quoc-Bao Nguyen, and John R Smith. Snap, eat, repeat: a food recognition engine for dietary logging. In *Proceedings of the 2nd international workshop on multimedia assisted dietary management*, 2016. 2

[32] Austin Meyers, Nick Johnston, Vivek Rathod, Anoop Korattikara, Alex Gorban, Nathan Silberman, Sergio Guadarrama, George Papandreou, Jonathan Huang, and Kevin P Murphy. Im2calories: towards an automated mobile vision food diary. In *ICCV*, 2015. 2

[33] Weiqing Min, Shuqiang Jiang, Shuhui Wang, Jitao Sang, and Shuhuan Mei. A delicious recipe analysis framework for exploring multi-modal recipes with various attributes. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 402–410, 2017. 2

[34] Weiqing Min, Linhu Liu, Zhiling Wang, Zhengdong Luo, Xiaoming Wei, Xiaolin Wei, and Shuqiang Jiang. Isia food-500: A dataset for large-scale food recognition via stacked global-local attention network. In *Proceedings of the 28th ACM International Conference on Multimedia*, 2020. 1, 2

[35] Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. Flow graph corpus from recipe texts. In *LREC*, pages 2370–2377, 2014. 2

[36] Taichi Nishimura, Suzushi Tomori, Hayato Hashimoto, Atsushi Hashimoto, Yoko Yamakata, Jun Harashima, Yoshitaka Ushiku, and Shinsuke Mori. Visual grounding annotation of recipe flow graph. In *LREC*, pages 4275–4284, 2020. 2

[37] Daniel Nyga and Michael Beetz. Everything robots always wanted to know about housework (but were afraid to ask). In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 243–250. IEEE, 2012. 2

[38] Dim P Papadopoulos, Youssef Tamaazousti, Ferda Ofli, Ingmar Weber, and Antonio Torralba. How to make a pizza: Learning a compositional layer-based gan model. In *CVPR*, 2019. 2

[39] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Training object class detectors with click supervision. In *CVPR*, 2017. 3

[40] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002. 7

[41] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *ICCV*, 2021. 8

[42] Hai X Pham, Ricardo Guerrero, Jiatong Li, and Vladimir Pavlovic. CHEF: Cross-modal Hierarchical Embeddings for Food Domain Retrieval. *AAAI*, 2021. 2, 5

[43] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *CVPR*, pages 8494–8502, 2018. 2

[44] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021. 8

[45] Nils Reimers, Iryna Gurevych, Nils Reimers, Iryna Gurevych, Nandan Thakur, Nils Reimers, Johannes Daxenberger, Iryna Gurevych, Nils Reimers, Iryna Gurevych, et al. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*, 2019. 3

[46] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 2015. 3, 6

[47] Amaia Salvador, Michal Drozdzal, Xavier Giro-i Nieto, and Adriana Romero. Inverse cooking: Recipe generation from food images. In *CVPR*, 2019. 1, 2

[48] Amaia Salvador, Erhan Gundogdu, Loris Bazzani, and Michael Donoser. Revamping cross-modal recipe retrieval with hierarchical transformers and self-supervised learning. In *CVPR*, 2021. 1, 2, 5, 6, 7

[49] Amaia Salvador, Nicholas Hynes, Yusuf Aytar, Javier Marin, Ferda Ofli, Ingmar Weber, and Antonio Torralba. Learning cross-modal embeddings for cooking recipes and food images. In *CVPR*, 2017. 1, 2, 4, 5, 6, 7, 8

[50] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics*, 1983. 7

[51] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015. 5

[52] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. In *NIPS*, 2004. 5

[53] Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2014. 5

[54] Alexander Sorokin and David Forsyth. Utility data annotation with amazon mechanical turk. In *Workshop at CVPR*, 2008. 3

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 4, 5, 6

[56] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *IJCV*, 2013. 3

[57] Hao Wang, Doyen Sahoo, Chenghao Liu, Ee-peng Lim, and Steven CH Hoi. Learning cross-modal embeddings with adversarial networks for cooking recipes and food images. In *CVPR*, 2019. 2, 5, 6, 7

[58] Hao Wang, Doyen Sahoo, Chenghao Liu, Ke Shu, Palakorn Achananuparp, Ee-peng Lim, and CH Steven Hoi. Cross-modal food retrieval: learning a joint embedding of food images and recipes with semantic consistency and attention mechanism. *IEEE Transactions on Multimedia*, 2021. 2, 5, 6

[59] Liwei Wang, Yin Li, and Svetlana Lazebnik. Learning deep structure-preserving image-text embeddings. In *CVPR*, 2016. 5

[60] Frank F Xu, Lei Ji, Botian Shi, Junyi Du, Graham Neubig, Yonatan Bisk, and Nan Duan. A benchmark for structured procedural knowledge extraction from cooking videos. In *NLP Beyond Text Workshop at EMNLP*, 2020. 2

[61] Yoko Yamakata, Shinsuke Mori, and John A Carroll. English recipe flow graph corpus. In *LREC*, pages 5187–5194, 2020. 2

[62] Yezhou Yang, Anupam Guha, Cornelia Fermüller, and Yiannis Aloimonos. Manipulation action tree bank: A knowledge resource for humanoids. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 987–992. IEEE, 2014. 2

[63] Yezhou Yang, Yi Li, Cornelia Fermuller, and Yiannis Aloimonos. Robot learning manipulation action plans by" watching" unconstrained videos from the world wide web. In *AAAI*, 2015. 2

[64] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B Tenenbaum. Neural-symbolic VQA: Disentangling reasoning from vision and language understanding. In *NeurIPS*, 2018. 2

[65] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Trans. on PAMI*, 2017. 3

[66] Bin Zhu and Chong-Wah Ngo. Cookgan: Causality based text-to-image synthesis. In *CVPR*, 2020. 2

[67] Bin Zhu, Chong-Wah Ngo, Jingjing Chen, and Yanbin Hao. R2gan: Cross-modal recipe retrieval with generative adversarial network. In *CVPR*, 2019. 1, 2, 5, 6, 7