Inverse Cooking: Recipe Generation from Food Images

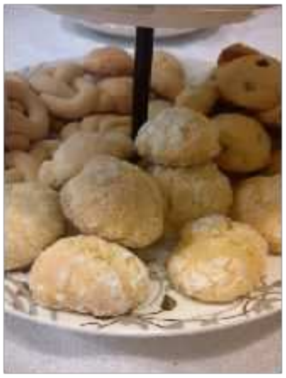
Amaia Salvador[1]* Michal Drozdzal[2] Xavier Giro-i-Nieto[1] Adriana Romero[2]
[1]Universitat Politecnica de Catalunya [2]Facebook AI Research
{amaia.salvador, xavier.giro}@upc.edu, {adrianars, mdrozdzal}@fb.com

Abstract

People enjoy food photography because they appreciate food. Behind each meal there is a story described in a complex recipe and, unfortunately, by simply looking at a food image we do not have access to its preparation process. Therefore, in this paper we introduce an inverse cooking system that recreates cooking recipes given food images. Our system predicts ingredients as sets by means of a novel architecture, modeling their dependencies without imposing any order, and then generates cooking instructions by attending to both image and its inferred ingredients simultaneously. We extensively evaluate the whole system on the large-scale Recipe1M dataset and show that (1) we improve performance w.r.t. previous baselines for ingredient prediction; (2) we are able to obtain high quality recipes by leveraging both image and ingredients; (3) our system is able to produce more compelling recipes than retrieval-based approaches according to human judgment. We make code and models publicly available[1].

1. Introduction

Food is fundamental to human existence. Not only does it provide us with energy—it also defines our identity and culture [10, 34]. As the old saying goes, *we are what we eat*, and food related activities such as cooking, eating and talking about it take a significant portion of our daily life. Food culture has been spreading more than ever in the current digital era, with many people sharing pictures of food they are eating across social media [31]. Querying Instagram for #food leads to at least 300M posts; similarly, searching for #foodie results in at least 100M posts, highlighting the unquestionable value that food has in our society. Moreover, eating patterns and cooking culture have been evolving over time. In the past, food was mostly prepared at home, but nowadays we frequently consume food prepared by third-parties (e.g. takeaways, catering and restaurants). Thus, the access to detailed information about prepared food is limited and, as a consequence, it is hard to know precisely what we eat. Therefore, we argue that there is a need for *inverse cooking* systems, which are able to infer ingredients and cooking instructions from a prepared meal.



Title: Biscuits

Ingredients:
Flour, butter, sugar, egg, milk, salt.

Instructions:
- Preheat oven to 450 degrees.
- Cream butter and sugar.
- Add egg and milk.
- Sift flour and salt together.
- Add to creamed mixture.
- Roll out on floured board to 1/4 inch thickness.
- Cut with biscuit cutter.
- Place on ungreased cookie sheet.
- Bake for 10 minutes.

Figure 1: **Example of a generated recipe**, composed of a title, ingredients and cooking instructions.

The last few years have witnessed outstanding improvements in visual recognition tasks such as natural image classification [47, 14], object detection [42, 41] and semantic segmentation [27, 19]. However, when comparing to natural image understanding, food recognition poses additional challenges, since food and its components have high intraclass variability and present heavy deformations that occur during the cooking process. Ingredients are frequently occluded in a cooked dish and come in a variety of colors, forms and textures. Further, visual ingredient detection requires high level reasoning and prior knowledge (e.g. *cake* will likely contain *sugar* and not *salt*, while *croissant* will presumably include *butter*). Hence, food recognition challenges current computer vision systems to go beyond the merely visible, and to incorporate prior knowledge to enable high-quality structured food preparation descriptions.

Previous efforts on food understanding have mainly focused on food and ingredient categorization [1, 39, 24]. However, a system for comprehensive visual food recognition should not only be able to recognize the type of meal or its ingredients, but also understand its preparation pro-

*Work done during internship at Facebook AI Research

[1] `https://github.com/facebookresearch/inversecooking`

cess. Traditionally, the image-to-recipe problem has been formulated as a retrieval task [54, 3, 4, 45], where a recipe is retrieved from a fixed dataset based on the image similarity score in an embedding space. The performance of such systems highly depends on the dataset size and diversity, as well as on the quality of the learned embedding. Not surprisingly, these systems fail when a matching recipe for the image query does not exist in the static dataset.

An alternative to overcome the dataset constraints of retrieval systems is to formulate the image-to-recipe problem as a conditional generation one. Therefore, in this paper, we present a system that *generates* a cooking recipe containing a title, ingredients and cooking instructions directly from an image. Figure 1 shows an example of a generated recipe obtained with our method, which first predicts ingredients from an image and then conditions on both the image and the ingredients to generate the cooking instructions. To the best of our knowledge, our system is the first to *generate* cooking recipes directly from food images. We pose the instruction generation problem as a sequence generation one *conditioned on two modalities* simultaneously, namely an image and its predicted ingredients. We formulate the ingredient prediction problem as a *set prediction*, exploiting their underlying structure. We model ingredient dependencies while not penalizing for prediction order, thus revising the question of *whether order matters* [51]. We extensively evaluate our system on the large-scale Recipe1M dataset [45] that contains images, ingredients and cooking instructions, showing satisfactory results. More precisely, in a human evaluation study, we show that our inverse cooking system outperforms previously introduced image-to-recipe retrieval approaches by a large margin. Moreover, using a small set of images, we show that food image-to-ingredient prediction is a hard task for humans and that our approach is able to surpass them.

The contributions of this paper can be summarized as:

– We present an inverse cooking system, which generates cooking instructions conditioned on an image and its ingredients, exploring different attention strategies to reason about both modalities simultaneously.

– We exhaustively study ingredients as both a *list* and a *set*, and propose a new architecture for ingredient prediction that exploits co-dependencies among ingredients without imposing order.

– By means of a user study we show that ingredient prediction is indeed a difficult task and demonstrate the superiority of our proposed system against image-to-recipe retrieval approaches.

2. Related Work

Food Understanding. The introduction of large scale food datasets, such as Food-101 [1] and Recipe1M [45], together with a recently held iFood challenge[2] has enabled significant advancements in visual food recognition, by providing reference benchmarks to train and compare machine learning approaches. As a result, there is currently a vast literature in computer vision dealing with a variety of food related tasks, with special focus in image classification [26, 39, 38, 33, 6, 24, 30, 60, 16, 17]. Subsequent works tackle more challenging tasks such as estimating the number of calories given a food image [32], estimating food quantities [5], predicting the list of present ingredients [3, 4] and finding the recipe for a given image [54, 3, 4, 45, 2]. Additionally, [34] provides a detailed cross-region analysis of food recipes, considering images, attributes (e.g. style and course) and recipe ingredients. Food related tasks have also been considered in the natural language processing literature, where recipe generation has been studied in the context of generating procedural text from either flow graphs [13, 36, 35] or ingredients' checklists [21].

Multi-label classification. Significant effort has been devoted in the literature to leverage deep neural networks for multi-label classification, by designing models [49, 8, 56, 37, 53] and studying loss functions [12] well suited for this task. Early attempts exploit single-label classification models coupled with binary logistic loss [3], assuming the independence among labels and dropping potentially relevant information. One way of capturing label dependencies is by relying on label powersets [49]. Powersets consider all possible label combinations, which makes them intractable for large scale problems. Another expensive alternative consists in learning the joint probability of the labels. To overcome this issue, probabilistic classifier chains [8] and their recurrent neural network-based [53, 37] counterparts propose to decompose the joint distribution into conditionals, at the expense of introducing intrinsic ordering. Note that most of these models require to make a prediction for each of the potential labels. Moreover, joint input and label embeddings [57, 25, 61] have been introduced to preserve correlations and predict label sets. As an alternative, researchers have attempted to predict the cardinality of the set of labels [43, 44]; however, assuming the independence of labels. When it comes to multi-label classification objectives, binary logistic loss [3], target distribution cross-entropy [12, 29], target distribution mean squared error [56] and ranking-based losses [12] have been investigated and compared. Recent results on large scale datasets outline the potential of the target distribution loss [29].

Conditional text generation. Conditional text generation with auto-regressive models has been widely studied in the literature using both text-based [48, 11, 50, 9] as well as image-based conditionings [52, 59, 28, 20, 23, 7, 46]. In neural machine translation, where the goal is to predict the translation for a given source text into another language, dif-

[2] https://www.kaggle.com/c/ifood2018

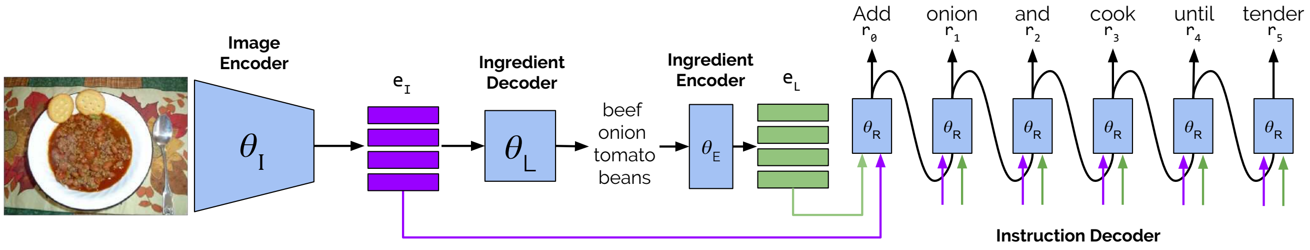


Figure 2: **Recipe generation model.** We extract image features $\mathbf{e}_I$ with the image encoder, parametrized by θ_I. Ingredients are predicted by θ_L, and encoded into ingredient embeddings $\mathbf{e}_L$ with θ_e. The cooking instruction decoder, parametrized by θ_R generates a recipe title and a sequence of cooking steps by attending to image embeddings $\mathbf{e}_I$, ingredient embeddings $\mathbf{e}_L$, and previously predicted words $(r_0, ..., r_{t-1})$.

ferent architecture designs have been studied, including recurrent neural networks [48], convolutional models [11] and attention based approaches [50]. More recently, sequence-to-sequence models have been applied to more open-ended generation tasks, such as poetry [55] and story generation [23, 9]. Following neural machine translation trends, autoregressive models have exhibited promising performance in image captioning [52, 59, 28, 20, 7, 46], where the goal is to provide a short description of the image contents, opening the doors to less constrained problems such as generating descriptive paragraphs [23] or visual storytelling [18].

3. Generating recipes from images

Generating a recipe (title, ingredients and instructions) from an image is a challenging task, which requires a simultaneous understanding of the ingredients composing the dish as well as the transformations they went through, e.g. slicing, blending or mixing with other ingredients. Instead of obtaining the recipe from an image directly, we argue that a recipe generation pipeline would benefit from an intermediate step predicting the ingredients list. The sequence of instructions would then be generated conditioned on both the image and its corresponding list of ingredients, where the interplay between image and ingredients could provide additional insights on how the latter were processed to produce the resulting dish.

Figure 2 illustrates our approach. Our recipe generation system takes a food image as an input and outputs a sequence of cooking instructions, which are generated by means of an instruction decoder that takes as input two embeddings. The first one represents visual features extracted from an image, while the second one encodes the ingredients extracted from the image. We start by introducing our transfomer-based instruction decoder in Subsection 3.1. This allows us to formally review the transformer, which we then study and modify to predict ingredients in an orderless manner in Subsection 3.2. Finally, we review the optimization details in Subsection 3.3.

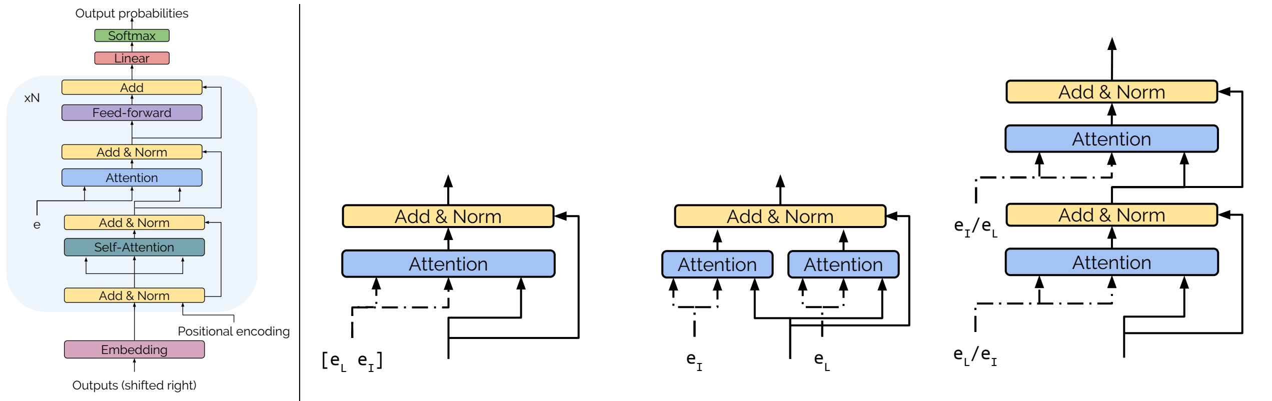
3.1. Cooking Instruction Transformer

Given an input image with associated ingredients, we aim to produce a sequence of instructions $R = (r_1, ..., r_T)$ (where r_t denotes a word in the sequence) by means of an instruction transformer [50]. Note that the title is predicted as the first instruction. This transformer is conditioned jointly on two inputs: the image representation $\mathbf{e}_I$ and the ingredient embedding $\mathbf{e}_L$. We extract the image representation with a ResNet-50 [15] encoder and obtain the ingredient embedding $\mathbf{e}_L$ by means of a decoder architecture to predict ingredients, followed by a single embedding layer mapping each ingredient into a fixed-size vector.

The instruction decoder is composed of *transformer blocks*, each of them containing two attention layers followed by a linear layer [50]. The first attention layer applies self-attention over previously generated outputs, whereas the second one attends to the model conditioning in order to refine the self-attention output. The transformer model is composed of multiple transformer blocks followed by a linear layer and a softmax nonlinearity that provides a distribution over recipe words for each time step t. Figure 3a illustrates the transformer model, which traditionally is conditioned on a single modality. However, our recipe generator is conditioned on two sources: the image features $\mathbf{e}_I \in \mathbb{R}^{P \times d_e}$ and ingredients embeddings $\mathbf{e}_L \in \mathbb{R}^{K \times d_e}$ (P and K denote the number of image and ingredient features, respectively, and d_e is the embedding dimensionality). Thus, we want our attention to reason about both modalities simultaneously, guiding the instruction generation process. To that end, we explore three different fusion strategies (depicted in Figure 3):

– **Concatenated attention.** This strategy first concatenates both image $\mathbf{e}_I$ and ingredients $\mathbf{e}_L$ embeddings over the first dimension $\mathbf{e}_{concat} \in \mathbb{R}^{(K+P) \times d_e}$. Then, attention is applied over the combined embeddings.

– **Independent attention.** This strategy incorporates two attention layers to deal with the bi-modal conditioning. In this case, one layer attends over the image embedding $\mathbf{e}_I$, whereas the other attends over the in-



(a) Transformer model [50] (b) Concatenated (c) Independent (d) Sequential

Figure 3: **Attention strategies for the instruction decoder.** In our experiments, we replace the attention module in the transformer (a), with three different attention modules (b-d) for cooking instruction generation using multiple conditions.

gredient embeddings $\mathbf{e}_L$. The output of both attention layers is combined via summation operation.

– **Sequential attention.** This strategy sequentially attends over the two conditioning modalities. In our design, we consider two orderings: (1) *image first* where the attention is first computed over image embeddings $\mathbf{e}_I$ and then over ingredient embeddings $\mathbf{e}_L$; and (2) *ingredients first* where the order is flipped and we first attend over ingredient embeddings $\mathbf{e}_L$ followed by image embeddings $\mathbf{e}_I$.

3.2. Ingredient Decoder

Which is the best structure to represent ingredients? On the one hand, it seems clear that ingredients are a *set*, since permuting them does not alter the outcome of the cooking recipe. On the other hand, we colloquially refer to ingredients as a *list* (e.g. list of ingredients), implying some order. Moreover, it would be reasonable to think that there is some information in the order in which humans write down the ingredients in a recipe. Therefore, in this subsection we consider both scenarios and introduce models that work either with a list of ingredients or with a set of ingredients.

A *list of ingredients* is a variable sized, ordered collection of unique meal constituents. More precisely, let us define a dictionary of ingredients of size N as $\mathcal{D} = \{d_i\}_{i=0}^N$, from which we can obtain a list of ingredients L by selecting K elements from $\mathcal{D}$: $L = [l_i]_{i=0}^K$. We encode L as a binary matrix $\mathbf{L}$ of dimensions $K \times N$, with $\mathbf{L}_{i,j} = 1$ if $d_j \in \mathcal{D}$ is selected and 0 otherwise (one-hot-code representation). Thus, our training data consists of M image and ingredient list pairs $\{(\mathbf{x}^{(i)}, \mathbf{L}^{(i)})\}_{i=0}^M$. In this scenario, the goal is to predict $\hat{\mathbf{L}}$ from an image $\mathbf{x}$ by maximizing the following objective:

$$\arg\max_{\theta_I,\theta_L} \sum_{i=0}^{M} \log p(\hat{\mathbf{L}}^{(i)} = \mathbf{L}^{(i)} | \mathbf{x}^{(i)}; \theta_I, \theta_L), \qquad (1)$$

where θ_I and θ_L represent the learnable parameters of the image encoder and ingredient decoder, respectively. Since $\mathbf{L}$ denotes a list, we can factorize $p(\hat{\mathbf{L}}^{(i)} = \mathbf{L}^{(i)} | \mathbf{x}^{(i)})$ into K conditionals: $\sum_{k=0}^{K} \log p(\hat{\mathbf{L}}_k^{(i)} = \mathbf{L}_k^{(i)} | \mathbf{x}^{(i)}, \mathbf{L}_{<k}^{(i)})$ [3] and parametrize $p(\hat{\mathbf{L}}_k^{(i)} | \mathbf{x}^{(i)}, \mathbf{L}_{<k}^{(i)})$ as a categorical distribution. In the literature, these conditionals are usually modeled with auto-regressive (recurrent) models. In our experiments, we choose the transformer model as well. It is worth mentioning that a potential drawback of this formulation is that it inherently penalizes for order, which might not necessarily be relevant for ingredients.

A *set of ingredients* is a variable sized, unordered collection of unique meal constituents. We can obtain a set of ingredients S by selecting K ingredients from the dictionary $\mathcal{D}$: $S = \{s_i\}_{i=0}^K$. We represent S as a binary vector $\mathbf{s}$ of dimension N, where $\mathbf{s}_i = 1$ if $\mathbf{s}_i \in S$ and 0 otherwise. Thus, our training data consists of M image and ingredient set pairs: $\{(\mathbf{x}^{(i)}, \mathbf{s}^{(i)})\}_{i=0}^M$. In this case, the goal is to predict $\hat{\mathbf{s}}$ from an image $\mathbf{x}$ by maximizing the following objective:

$$\arg\max_{\theta_I,\theta_L} \sum_{i=0}^{M} \log p(\hat{\mathbf{s}}^{(i)} = \mathbf{s}^{(i)} | \mathbf{x}^{(i)}; \theta_I, \theta_L). \qquad (2)$$

Assuming independence among elements, we can factorize $p(\hat{\mathbf{s}}^{(i)} = \mathbf{s}^{(i)} | \mathbf{x}^{(i)})$ as $\sum_{j=0}^{N} \log p(\hat{\mathbf{s}}_j^{(i)} = \mathbf{s}_j^{(i)} | \mathbf{x}^{(i)})$. However, the ingredients in the set are not necessarily independent, e.g. *salt* and *pepper* frequently appear together.

To account for element dependencies in the set, we model the set as a list, i.e. as a product of conditional probabilities, by means of an auto-regressive model such as the transformer. The transformer predicts ingredients in a list-like fashion $p(\hat{\mathbf{L}}_k^{(i)} | \mathbf{x}^{(i)}, \mathbf{L}_{<k}^{(i)})$, until the end of sequence *eos* token is encountered. As mentioned previously, the drawback of this approach is that such model design *penalizes*

[3] $\mathbf{L}_k^{(i)}$ denotes the k-th row of $\mathbf{L}^{(i)}$ and $\mathbf{L}_{<k}^{(i)}$ represents all rows of $\mathbf{L}^{(i)}$ up to, but not including, the k-th one.

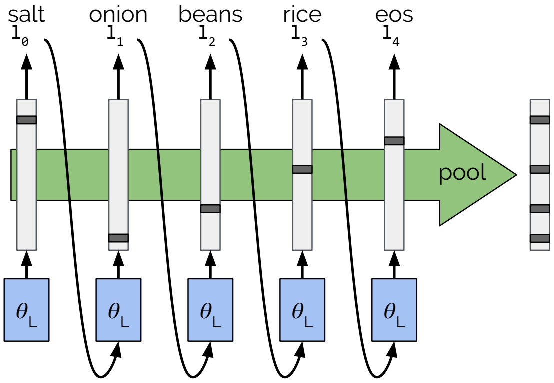


Figure 4: **Set transformer** (TF_{set})**.** Softmax probabilities are pooled across time to avoid penalizing for order.

for order. In order to remove the order in which ingredients are predicted, we propose to aggregate the outputs across different time-steps by means of a max pooling operation (see Figure 4). Moreover, to ensure that the ingredients in $\hat{\mathbf{L}}^{(i)}$ are selected without repetition, we force the pre-activation of $p(\hat{\mathbf{L}}_k^{(i)}|\mathbf{x}^{(i)}, \mathbf{L}_{<k}^{(i)})$ to be $-\infty$ for all previously selected ingredients at time-steps $< k$. We train this model by minimizing the binary cross-entropy between the predicted ingredients (after pooling) and the ground truth. Including the *eos* in the pooling operation would result in loosing the information of where the token appears. Therefore, in order to learn the stopping criteria of the ingredient prediction, we introduce an additional loss accounting for it. The *eos* loss is defined as the binary cross-entropy loss between the predicted *eos* probability at all time-steps and the ground truth (represented as a unit step function, whose value is 0 for the time-steps corresponding to ingredients and 1 otherwise). In addition to that, we incorporate a cardinality ℓ_1 penalty, which we found empirically useful. At inference time, we directly sample from the transformer's output. We refer to this model as *set transformer*.

Alternatively, we could use *target distribution* $p(\mathbf{s}^{(i)}|\mathbf{x}^{(i)}) = \mathbf{s}^{(i)}/\sum_j \mathbf{s}_j^{(i)}$ [12, 29] to model the joint distribution of set elements and train a model by minimizing the cross-entropy loss between $p(\mathbf{s}^{(i)}|\mathbf{x}^{(i)})$ and the model's output distribution $p(\hat{\mathbf{s}}^{(i)}|\mathbf{x}^{(i)})$. Nonetheless, it is not clear how to convert the target distribution back to the corresponding set of elements with variable cardinality. In this case, we build a feed forward network and train it with the target distribution cross-entropy loss. To recover the ingredient set, we propose to greedily sample elements from a *cumulative distribution of sorted output probabilities* $p(\hat{\mathbf{s}}^{(i)}|\mathbf{x}^{(i)})$ and stop the sampling once the sum of probabilities of selected elements is above a threshold. We refer to this model as *feed forward (target distribution)*.

3.3. Optimization

We train our recipe transfomer in two stages. In the first stage, we pre-train the image encoder and ingredients decoder as presented in Subsection 3.2. Then, in the second stage, we train the ingredient encoder and instruction decoder (following Subsection 3.1) by minimizing the negative log-likelihood and adjusting θ_R and θ_E. Note that, while training, the instruction decoder takes as input the ground truth ingredients. All transformer models are trained with teacher forcing [58] except for the set transformer.

4. Experiments

This section is devoted to the dataset and the description of implementation details, followed by an exhaustive analysis of the proposed attention strategies for the cooking instruction transformer. Further, we quantitatively compare the proposed ingredient prediction models to previously introduced baselines. Finally, a comparison of our inverse cooking system with retrieval-based models as well as a comprehensive user study is provided.

4.1. Dataset

We train and evaluate our models on the Recipe1M dataset [45], composed of 1 029 720 recipes scraped from cooking websites. The dataset contains 720 639 training, 155 036 validation and 154 045 test recipes, containing a title, a list of ingredients, a list of cooking instructions and (optionally) an image. In our experiments, we use only the recipes containing images, and remove recipes with less than 2 ingredients or 2 instructions, resulting in 252 547 training, 54 255 validation and 54 506 test samples.

Since the dataset was obtained by scraping cooking websites, the resulting recipes are highly unstructured and contain frequently redundant or very narrowly defined cooking ingredients (e.g. *olive oil*, *virgin olive oil* and *spanish olive oil* are separate ingredients). Moreover, the ingredient vocabulary contains more than 400 different types of *cheese*, and more than 300 types of *pepper*. As a result, the original dataset contains 16 823 unique ingredients, which we preprocess to reduce its size and complexity. First, we merge ingredients if they share the first or last two words (e.g. *bacon cheddar cheese* is merged into *cheddar cheese*); then, we cluster the ingredients that have same word in the first or in the last position (e.g. *gorgonzola cheese* or *cheese blend* are clustered together into the *cheese* category); finally we remove plurals and discard ingredients that appear less than 10 times in the dataset. Altogether, we reduce the ingredient vocabulary from over $16k$ to 1 488 unique ingredients. For the cooking instructions, we tokenize the raw text and remove words that appear less than 10 times in the dataset, and replace them with unknown word token. Moreover, we add special tokens for the start and the end of recipe as well

Model	ppl
Independent	8.59
Seq. img. first	8.53
Seq. ing. first	8.61
Concatenated	**8.50**

Model	IoU	F1
FF_{BCE}	17.85	30.30
FF_{IOU}	26.25	41.58
FF_{DC}	27.22	42.80
FF_{TD}	**28.84**	**44.11**
TF_{list}	29.48	45.55
TF_{list} + shuf.	27.86	43.58
TF_{set}	**31.80**	**48.26**

Table 1: **Model selection (val).** Left: Recipe perplexity (ppl). Right: Global ingredient IoU & F1.

as the end of instruction. This process results in a recipe vocabulary of 23 231 unique words.

4.2. Implementation Details

We resize images to 256 pixels in their shortest side and take random crops of 224 × 224 for training and we select central 224 × 224 pixels for evaluation. For the instruction decoder, we use a transformer with 16 blocks and 8 multi-head attentions, each one with dimensionality 64. For the ingredient decoder, we use a transformer with 4 blocks and 2 multi-head attentions, each one with dimensionality of 256. To obtain image embeddings we use the last convolutional layer of ResNet-50 model. Both image and ingredients embedings are of dimension 512. We keep a maximum of 20 ingredients per recipe and truncate instructions to a maximum of 150 words. The models are trained with Adam optimizer [22] until early-stopping criteria is met (using patience of 50 and monitoring validation loss). All models are implemented with PyTorch[4] [40]. Additional implementation details are provided in the supplementary material.

4.3. Recipe Generation

In this section, we compare the proposed multi-modal attention architectures described in Section 3.1. Table 1 (left) reports the results in terms of perplexity on the validation set. We observe that independent attention exhibits the lowest results, followed by both sequential attentions. While the latter have the capability to refine the output with either ingredient or image information consecutively, independent attention can only do it in one step. This is also the case of concatenated attention, which achieves the best performance. However, concatenated attention is flexible enough to decide whether to give more focus to one modality, at the expense of the other, whereas independent attention is forced to include information from both modalities. Therefore, we use the concatenated attention model to report results on the test set. We compare it to a system going directly from image-to-sequence of instructions without predicting ingredients (I2R). Moreover, to assess the influence of visual features on recipe quality, we adapt our model by removing visual features and predicting instructions directly from ingredients (L2R). Our system achieves a test set perplexity of 8.51, improving both I2R and L2R baselines, and highlighting the benefits of using both image and ingredients when generating recipes. L2R surpasses I2R with a perplexity of 8.67 vs. 9.66, demonstrating the usefulness of having access to concepts (ingredients) that are essential to the cooking instructions. Finally, we greedily sample instructions from our model and analyze the results. We notice that generated instructions have an average of 9.21 sentences containing 9 words each, whereas real, ground truth instructions have an average of 9.08 sentences of length 12.79. See supplementary material for qualitative examples of generated recipes.

4.4. Ingredient Prediction

In this section, we compare the proposed ingredient prediction approaches to previously introduced models, with the goal of assessing whether ingredients should be treated as lists or sets. We consider models from the multilabel classification literature as baselines, and tune them for our purposes. On the one hand, we have models based on feed forward convolutional networks, which are trained to predict sets of ingredients. We experiment with several losses to train these models, namely binary cross-entropy, soft intersection over union as well as target distribution cross-entropy. Note that binary cross-entropy is the only one not taking into account dependencies among elements in the set. On the other hand, we have sequential models that predict lists, imposing order and exploiting dependencies among elements. Finally, we consider recently proposed models which couple set prediction with cardinality prediction to determine which elements to include in the set [44].

Table 1 (right) reports the results on the validation set for the state-of-the-art baselines as well as the proposed approaches. We evaluate the models in terms of Intersection over Union (IoU) and F1 score, computed for accumulated counts of TP, FN and FP over the entire dataset split (following Pascal VOC convention). As shown in the table, the feed forward model trained with binary cross-entropy [3] (FF_{BCE}) exhibits the lowest performance on both metrics, which could be explained by the assumed independence among ingredients. These results are already notably improved by the method that learns to predict the set cardinality (FF_{DC}). Similarly, the performance increases when training the model with structured losses such as soft IoU (FF_{IOU}). Our feed forward model trained with target distribution (FF_{TD}) and sampled by thresholding (th = 0.5) the sum of probabilities of selected ingredients outperforms all feed forward baselines, including recently proposed alternatives for set prediction such as [44] (FF_{DC}). Note that target distribution models dependencies among

[4] https://pytorch.org/

	Card. error	# pred. ingrs
FF_{BCE}	5.67 ± 3.10	2.37 ± 1.58
FF_{DC}	2.68 ± 2.07	9.18 ± 2.06
FF_{IOU}	2.46 ± 1.95	7.86 ± 1.72
FF_{TD}	3.02 ± 2.50	8.02 ± 3.24
TF_{list}	2.49 ± 2.11	7.05 ± 2.77
TF_{list} + shuffle	3.24 ± 2.50	5.06 ± 1.85
TF_{set}	2.56 ± 1.93	9.43 ± 2.35

Table 2: **Ingredient Cardinality.**

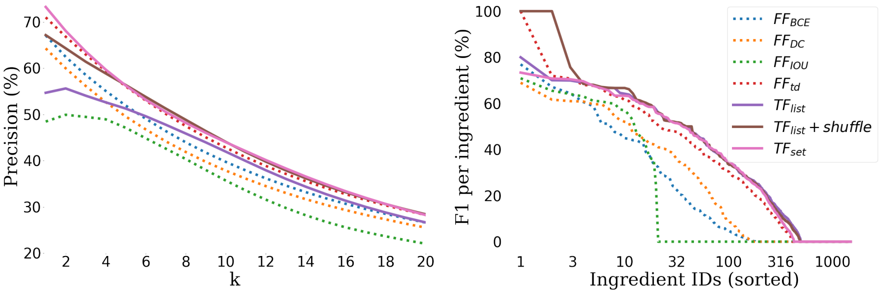


Figure 5: **Ingredient prediction results:** P@K and F1 per ingredient.

elements in a set and implicitly captures cardinality information. Following recent literature modeling sets as lists [37], we train a transformer network to predict ingredients given an image by minimizing the negative log-likelihood loss (TF_{list}). Moreover, we train the same transformer by randomly shuffling the ingredients (thus, removing order from the data). Both models exhibit competitive results when compared to feed forward models, highlighting the importance of modeling dependencies among ingredients. Finally, our proposed set transformer TF_{set}, which models ingredient co-occurrences exploiting the auto-regressive nature of the model yet satisfying order invariance, achieves the best results, emphasizing the importance of modeling dependencies, while not penalizing for any given order.

The average number of ingredients per sample in Recipe1M is 7.99 ± 3.21 after pre-processing. We report the cardinality prediction errors as well as the average number of predicted ingredients for each of the tested models in Table 2. TF_{set} is the third best method in terms of cardinality error (after FF_{IOU} and TF_{list}), while being superior to all methods in terms of F1 and IoU. Further, Figure 5 (left) shows the precision score at different values of K. As observed, the plot follows similar trends as Table 1 (right), with FF_{TD} being among the most competitive models and TF_{set} outperforming all previous baselines for most values of K. Figure 5 (right) shows the F1 per ingredient, where the ingredients in the horizontal axes are sorted by score. Again, we see that models that exploit dependencies consistently improve ingredient's F1 scores, strengthening the importance of modeling ingredient co-occurrences.

4.5. Generation vs Retrieval

In this section, we compare our proposed recipe generation system with retrieval baselines, which we use to search recipes in the *entire* test set for fair comparison.

Ingredient prediction evaluation. We use the retrieval model in [45] as a baseline and compare it with our best ingredient predictions models, namely FF_{TD} and FF_{set}. The retrieval model, which we refer to as R_{I2LR}, learns joint embeddings of images and recipes (title, ingredients and instructions). Therefore, for the ingredient prediction task, we use the image embeddings to retrieve the closest recipe and report metrics for the ingredients of the retrieved recipe. We further consider an alternative retrieval architecture, which learns joint embeddings between images and ingredients list (ignoring title and instructions). We refer to this model as R_{I2L}. Table 3 (left) reports the obtained results on the Recipe1M test set. The R_{I2LR} model outperforms the R_{I2L} one, which indicates that instructions contain complementary information that is useful when learning effective embeddings. Furthermore, both of our proposed methods outperform the retrieval-baselines by a large margin (e.g. TF_{set} outperforms the R_{I2LR} retrieval baseline by 12.26 IoU points and 15.48 F1 score points), which demonstrates the superiority of our models. Finally, Figure 6 presents some qualitative results for image-to-ingredient prediction for our model as well as for the retrieval based system. We use blue to highlight the ingredients that are present in the ground truth annotation and red otherwise.

	IoU	F1
R_{I2L} [45]	18.92	31.83
R_{I2LR} [45]	19.85	33.13
FF_{TD} (ours)	29.82	45.94
TF_{set} (ours)	**32.11**	**48.61**

	Rec.	Prec.
R_{IL2R}	31.92	28.94
Ours	**75.47**	**77.13**

Table 3: **Test performance against retrieval.** Left: Global ingredient IoU and F1 scores. Right: Precision and Recall of ingredients in cooking instructions.

	IoU	F1
Human	21.36	35.20
Retrieved	18.03	30.55
Ours	**32.52**	**49.08**

	Success %
Real	80.33
Retrieved [45]	48.81
Ours	55.47

Table 4: **User studies.** Left: IoU & F1 scores for ingredients obtained by retrieval system, our approach and humans. Right: Recipe success rate according to human judgment.

Recipe generation evaluation. We compare our proposed instruction decoder (which generates instructions given an image and ingredients) with a retrieval variant. For

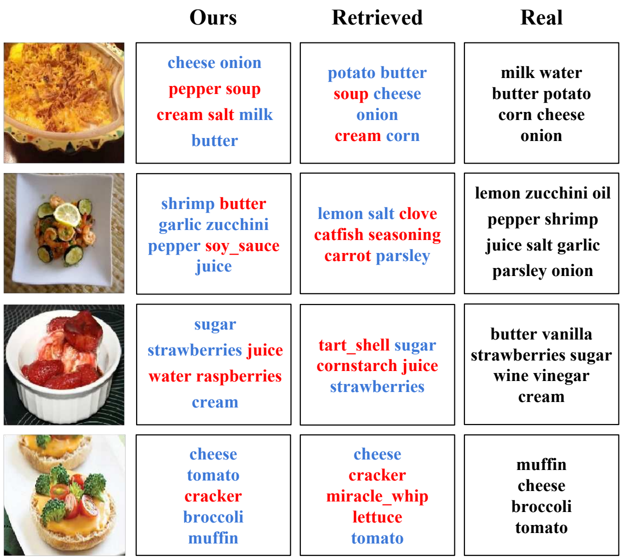

Ours
Retrieved
Real
cheese onion pepper soup cream salt milk butter
potato butter soup cheese onion cream corn
milk water butter potato corn cheese onion
shrimp butter garlic zucchini pepper soy_sauce juice
lemon salt clove catfish seasoning carrot parsley
lemon zucchini oil pepper shrimp juice salt garlic parsley onion
sugar strawberries juice water raspberries cream
tart_shell sugar cornstarch juice strawberries
butter vanilla strawberries sugar wine vinegar cream
cheese tomato cracker broccoli muffin
cheese cracker miracle_whip lettuce tomato
muffin cheese broccoli tomato

Figure 6: **Ingredient prediction examples.** We compare obtained ingredients with our method and the retrieval baseline. Ingredients are displayed in blue if they are present in the real sample and red otherwise. Best viewed in color.

a fair comparison, we retrain the retrieval system to find the cooking instructions given both image and ingredients. In our evaluation, we consider the ground truth ingredients as reference and compute *recall* and *precision* w.r.t. the ingredients that appear in the obtained instructions. Thus, recall computes the percentage of ingredients in the reference that appear in the output instructions, whereas precision measures the percentage of ingredients appearing in the instructions that also appear in the reference. Table 3 (right) displays comparison between our model and the retrieval system. Results show that ingredients appearing in generated instructions have better recall and precision scores than the ingredients in retrieved instructions.

4.6. User Studies

In this section, we quantify the quality of predicted ingredients and generated instructions with user studies. In the first study, we compare the performance of our model against human performance in the task of recipe generation (including ingredients and recipe instructions). We randomly select 15 images from the test set, and ask users to select up to 20 distinct ingredients as well as write a recipe that would correspond with the provided image. To reduce the complexity of the task for humans, we reduced the ingredient vocabulary from 1 488 to 323, by increasing the frequency threshold from 10 to 1k. We collected answers from 31 different users, altogether collecting an average of 5.5 answers for each image. For fair comparison, we re-train our best ingredient prediction model on the reduced vocabulary of ingredients. We compute IoU and F1 ingredient scores obtained by humans, the retrieval baseline and our method. Results are included in Table 4 (left), underlining the complexity of the task. As shown in the table, humans outperform the retrieval baseline (F1 of 35.20% vs 30.55%, respectively). Furthermore, our method outperforms both human baseline and retrieval based systems obtaining F1 of 49.08%. Qualitative comparisons between generated and human-written recipes (including recipes from average and expert users) are provided in the supplementary material.

The second study aims at quantifying the quality of the generated recipes (ingredients and instructions) with respect to (1) the real recipes in the dataset, and (2) the ones obtained with the retrieval baseline [45]. With this purpose, we randomly select 150 recipes with their associated images from the test set and, for each image, we collect the corresponding real recipe, the top-1 retrieved recipe and our generated recipe. We present the users with 15 image-recipe pairs (randomly chosen among the real, retrieved and generated ones) asking them to indicate whether the recipe matches the image. In the study, we collected answers from 105 different users, resulting in an average of 10 responses for each image. Table 4 (right) presents the results of this study, reporting the success rate of each recipe type. As it can be observed, the success rate of generated recipes is higher than the success rate of retrieved recipes, stressing the benefits of our approach w.r.t. retrieval.

5. Conclusion

In this paper, we introduced an image-to-recipe generation system, which takes a food image and produces a recipe consisting of a title, ingredients and sequence of cooking instructions. We first predicted sets of ingredients from food images, showing that modeling dependencies matters. Then, we explored instruction generation conditioned on images and inferred ingredients, highlighting the importance of reasoning about both modalities at the same time. Finally, user study results confirm the difficulty of the task, and demonstrate the superiority of our system against state-of-the-art image-to-recipe retrieval approaches.

6. Acknowledgements

We are grateful to Nicolas Ballas, Lluis Castrejon, Zizhao Zhang and Pascal Vincent for their fruitful comments and suggestions. We also want to express our gratitude to Joelle Pineau for her unwavering support to this project. Finally, we wish to thank everyone who anonymously participated in the user studies.

This work has been partially developed in the framework of projects TEC2013-43935-R and TEC2016-75976-R, financed by the Spanish Ministerio de Economa y Competitividad and the European Regional Development Fund.

References

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

[2] 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.

[3] Jing-Jing Chen and Chong-Wah Ngo. Deep-based ingredient recognition for cooking recipe retrieval. In *ACM Multimedia*. ACM, 2016.

[4] Jing-Jing Chen, Chong-Wah Ngo, and Tat-Seng Chua. Cross-modal recipe retrieval with rich food attributes. In *ACM Multimedia*. ACM, 2017.

[5] Mei-Yun Chen, Yung-Hsiang Yang, Chia-Ju Ho, Shih-Han Wang, Shane-Ming Liu, Eugene Chang, Che-Hua Yeh, and Ming Ouhyoung. Automatic chinese food identification and quantity estimation. In *SIGGRAPH Asia 2012 Technical Briefs*, 2012.

[6] Xin Chen, Hua Zhou, and Liang Diao. Chinesefoodnet: A large-scale image dataset for chinese food recognition. *CoRR*, abs/1705.02743, 2017.

[7] Bo Dai, Dahua Lin, Raquel Urtasun, and Sanja Fidler. Towards diverse and natural image descriptions via a conditional gan. *ICCV*, 2017.

[8] Krzysztof Dembczyński, Weiwei Cheng, and Eyke Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, 2010.

[9] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *ACL*, 2018.

[10] Claude Fischler. Food, self and identity. *Information (International Social Science Council)*, 1988.

[11] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122, 2017.

[12] Yunchao Gong, Yangqing Jia, Thomas Leung, Alexander Toshev, and Sergey Ioffe. Deep convolutional ranking for multilabel image annotation. *CoRR*, abs/1312.4894, 2013.

[13] Kristian J. Hammond. CHEF: A model of case-based planning. In *AAAI*, 1986.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *CVPR*, 2015.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[16] Luis Herranz, Shuqiang Jiang, and Ruihan Xu. Modeling restaurant context for food recognition. *IEEE Transactions on Multimedia*, 2017.

[17] Shota Horiguchi, Sosuke Amano, Makoto Ogawa, and Kiyoharu Aizawa. Personalized classifier for food image recognition. *IEEE Transactions on Multimedia*, 2018.

[18] Qiuyuan Huang, Zhe Gan, Asli Çelikyilmaz, Dapeng Oliver Wu, Jianfeng Wang, and Xiaodong He. Hierarchically structured reinforcement learning for topically coherent visual story generation. *CoRR*, abs/1805.08191, 2018.

[19] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *CVPR-W*, 2017.

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

[21] Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. Globally coherent text generation with neural checklist models. In *EMNLP*, 2016.

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[23] Jonathan Krause, Justin Johnson, Ranjay Krishna, and Li Fei-Fei. A hierarchical approach for generating descriptive image paragraphs. In *CVPR*, 2017.

[24] Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. In *CVPR*, 2018.

[25] Zijia Lin, Guiguang Ding, Mingqing Hu, and Jianmin Wang. Multi-label classification via feature-aware implicit label space encoding. In *ICML*, 2014.

[26] 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 *ICOST*, 2016.

[27] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[28] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *CVPR*, 2017.

[29] Dhruv Mahajan, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. *CoRR*, abs/1805.00932, 2018.

[30] Niki Martinel, Gian Luca Foresti, and Christian Micheloni. Wide-slice residual networks for food recognition. In *WACV*, 2018.

[31] Sara McGuire. Food Photo Frenzy: Inside the Instagram Craze and Travel Trend. `https://www.business.com/articles/food-photo-frenzy-inside-the-instagram-craze-and-travel-trend/`, 2017. [Online; accessed Nov-2018].

[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.

[33] Simon Mezgec and Barbara Koroui Seljak. Nutrinet: A deep learning food and drink image recognition system for dietary assessment. *Nutrients*, 9(7), 2017.

[34] Weiqing Min, Bing-Kun Bao, Shuhuan Mei, Yaohui Zhu, Yong Rui, and Shuqiang Jiang. You are what you eat: Exploring rich recipe information for cross-region food analysis. *IEEE Transactions on Multimedia*, 2018.

[35] Shinsuke Mori, Hirokuni Maeta, Tetsuro Sasada, Koichiro Yoshino, Atsushi Hashimoto, Takuya Funatomi, and Yoko Yamakata. Flowgraph2text: Automatic sentence skeleton

compilation for procedural text generation. In *INLG*. The Association for Computer Linguistics, 2014.

[36] Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. Flow graph corpus from recipe texts. In *LREC*. European Language Resources Association (ELRA), 2014.

[37] Jinseok Nam, Eneldo Loza Mencía, Hyunwoo J Kim, and Johannes Fürnkranz. Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *NeurIPS*. 2017.

[38] Chong-Wah Ngo. Deep learning for food recognition. In *SoICT*, 2017.

[39] Ferda Ofli, Yusuf Aytar, Ingmar Weber, Raggi al Hammouri, and Antonio Torralba. Is saki# delicious?: The food perception gap on instagram and its relation to health. In *ICWWW*, 2017.

[40] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NeurIPS-W*, 2017.

[41] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[42] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[43] S Hamid Rezatofighi, Anton Milan, Ehsan Abbasnejad, Anthony Dick, Ian Reid, et al. Deepsetnet: Predicting sets with deep neural networks. In *ICCV*, 2017.

[44] S Hamid Rezatofighi, Anton Milan, Qinfeng Shi, Anthony Dick, and Ian Reid. Joint learning of set cardinality and state distribution. *AAAI*, 2018.

[45] 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. *CVPR*, 2017.

[46] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *ACL*, 2018.

[47] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[48] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NeurIPS*, 2014.

[49] Grigorios Tsoumakas and Ioannis Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In Joost N. Kok, Jacek Koronacki, Raomon Lopez de Mantaras, Stan Matwin, Dunja Mladenič, and Andrzej Skowron, editors, *ECML*, 2007.

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

[51] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. In *ICLR*, 2016.

[52] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015.

[53] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. CNN-RNN: A unified framework for multi-label image classification. In *CVPR*, 2016.

[54] Xin Wang, Devinder Kumar, Nicolas Thome, Matthieu Cord, and Frederic Precioso. Recipe recognition with large multimodal food dataset. In *ICMEW*, 2015.

[55] Zhe Wang, Wei He, Hua Wu, Haiyang Wu, Wei Li, Haifeng Wang, and Enhong Chen. Chinese poetry generation with planning based neural network. *CoRR*, abs/1610.09889, 2016.

[56] Yunchao Wei, Wei Xia, Junshi Huang, Bingbing Ni, Jian Dong, Yao Zhao, and Shuicheng Yan. CNN: single-label to multi-label. *CoRR*, abs/1406.5726, 2014.

[57] Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, 2011.

[58] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*, 1(2), June 1989.

[59] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[60] Ruihan Xu, Luis Herranz, Shuqiang Jiang, Shuang Wang, Xinhang Song, and Ramesh Jain. Geolocalized modeling for dish recognition. *IEEE Transactions on Multimedia*, 2015.

[61] Chih-Kuan Yeh, Wei-Chieh Wu, Wei-Jen Ko, and Yu-Chiang Frank Wang. Learning deep latent spaces for multi-label classification. *CoRR*, abs/1707.00418, 2017.