6-2020

Cookgan: Causality based text-to-image synthesis

Bin ZHU

Chong-wah NGO
Singapore Management University, cwngo@smu.edu.sg

Citation

ZHU, Bin and NGO, Chong-wah. Cookgan: Causality based text-to-image synthesis. (2020). *Proceedings of 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, June 13-19*. 5518-5526.
Available at: https://ink.library.smu.edu.sg/sis_research/6484

CookGAN: Causality based Text-to-Image Synthesis

Bin Zhu
City University of Hong Kong
binzhu4-c@my.cityu.edu.hk

Chong-Wah Ngo
City University of Hong Kong
cscwngo@cityu.edu.hk

Abstract

This paper addresses the problem of text-to-image synthesis from a new perspective, i.e., the cause-and-effect chain in image generation. Causality is a common phenomenon in cooking. The dish appearance changes depending on the cooking actions and ingredients. The challenge of synthesis is that a generated image should depict the visual result of action-on-object. This paper presents a new network architecture, CookGAN, that mimics visual effect in causality chain, preserves fine-grained details and progressively upsamples image. Particularly, a cooking simulator sub-network is proposed to incrementally make changes to food images based on the interaction between ingredients and cooking methods over a series of steps. Experiments on Recipe1M verify that CookGAN manages to generate food images with reasonably impressive inception score. Furthermore, the images are semantically interpretable and manipulable.

1. Introduction

Text-to-image synthesis aims to generate images from natural language description. A generated image is expected to be photo and semantics realistic. Specifically, an image should have sufficient visual details that semantically align with the text description. Since the proposal of Generative Adversarial Network (GAN) [1], there have been numerous progresses that address the issues in photo-realistic quality [11, 18, 19, 20], and semantic consistency [17]. While both aspects emphasize image quality, an aspect being overlooked in the literature is the cause-and-effect visual scenario in image generation. For example, an image corresponding to the text "cut chicken into dice and stir with roasted peanuts" is hard to be generated with the current text-to-image synthesis paradigm. The reason is that the sentence is action-oriented. The expected image details are entities like "diced chicken" and "roasted peanuts", and the visual consequence of stirring both entities. The current state-of-the-art techniques that rely on mapping between textual and visual entities cannot deal with this cause-and-effect realistic image generation.

This paper studies recipe-to-image synthesis, specifically, to generate food image from cooking recipe. Different from visual narrative sentences that describe the visual content expected in an image, recipes provide ingredients as entities and cooking steps to textually instruct preparation of a dish. The expected image is to present the final prepared dish as a visual consequence over a series of cooking steps. Note that a cooking step is not necessarily to be visually relevant. It implies a new state of ingredient entities after the step is taken.

We propose a new network architecture, named CookGAN, to address the causality effect in image generation. Different from other GANs [18, 19, 20, 16, 23, 17], CookGAN is a tailor-made GAN for food image generation. The input to CookGAN consists of a list of words (i.e., ingredients) and a sequence of procedural descriptions (i.e., cooking steps). CookGAN addresses four issues in generating causality realistic food image. First, the network allows explicit interaction between cooking steps and ingredients. Second, the evolution of dish over different steps is learnt incrementally, such that on-the-fly modification of ingredients and instructions is possible to visualize novel effect of a dish. Third, the bundled effect of ingredient-action can be modeled. For example, the shape of egg changes depending on whether an action is boil, fry or steam. Fourth, the visibility and impact of ingredients to dish are learnt. For example, "sugar" is likely non-visible, while "tomato sauce" can significantly change the appearance of a dish.

Other than these issues, photo and semantics realistic image generation are also considered. Similar in spirit as other GANs [19, 20, 16], CookGAN progressively upsamples image from low to high resolution. The relative importance of each ingredient to the sub-regions of a generated image is computed for presentation of fine-grained ingredient details. The ingredients of a generated food image can also be semantically interpreted using an existing ingredient encoder [13].

The main contribution of this paper is addressing causality effect in image generation. To the best of our knowledge, there is no prior work for this problem. We contribute a novel network, CookGAN, that vividly mimics cooking process in step-by-step learning of cause-and-effect scenario. CookGAN is capable of generating photo and causality realistic food images as demonstrated in the experiments.

2. Related Works

Conditional GAN (CGAN) [9] has pushed forward the rapid progress of text-to-image synthesis. By employing CGAN, Reed *et al.*[11] proposed a complete and standard pipeline of text-to-image synthesis to generate images from text descriptions. Nevertheless, the size of image is as small as 64 $\times$ 64. To address this problem, StackGAN [18] was proposed to generate higher resolution images in two stages. The CGAN in first stage captures the primitive shapes and basic colors of objects in a low-resolution image. Together with text descriptions, the image is refined with more details by the CGAN in the second stage.

Nevertheless, the two stages in StackGAN are learnt independently, resulting in lack of interaction between two closely related stages. This problem is addressed by training an end-to-end multi-stage GANs. Both StackGAN++ [19] and HDGAN [20] adopt tree-like structure with multiple pairs of generators and discriminators to progressively generate an image from low to high resolution. Different branches of the tree capture image distributions at various scales of resolution. The progressive way of upsampling enables a generator to inherit the mid-level features from the previous immediate stage for effective image generation. A limitation of these approaches, nevertheless, is that image generation is sensitive to different ways of expressing the meaning of a sentence. The issue of consistent semantics was recently addressed by SDGAN [17]. Consistent high-level semantics and diverse low-level semantics are guaranteed by a Siamese mechanism and semantic-conditioned batch normalization respectively.

The aforementioned approaches take only sentence-level features as condition. As a consequence, fine-grained image details are often missing in the generated images. This issue is addressed in AttnGAN [16] and DMGAN [23] by further leveraging word-level features as condition. By assigning weights to words, AttnGAN [16] manages to generate fine-grained image details at different regions of an image. Furthermore, the generator is trained by ensuring cross-modal similarity between a text description and the generated image. Similar in spirit but through a different way, DMGAN [23] employs a memory network to determine word importance based on the initial generated image. By fusing image content and word-attended text, an image is dynamically refined with fine-grained details.

CookGAN inherits most properties of the existing GANs for text-to-image synthesis. Specifically, CookGAN is multi-stage as [19, 20, 16] for progressive upsampling, and emphasizes word-level condition as [16] for fine-grained image generation. Different from these GANs, CookGAN is designed to address causality effect. Specially, the cause-and-effect evolution of a food image is implicitly captured in a sequential network, while fine-grained ingredient details are explicitly modeled with attention-like network.

There are few works [4, 6, 2, 21, 10] addressing the problem of food image synthesis. In [4], rather than using recipe, image is generated based on food style. In [6, 2], only ingredients are exploited for food image generation while cooking steps are ignored. PizzaGAN is proposed in [10] to synthesize food images by learning different operators to add, remove and cook ingredients. Different from CookGAN where input is recipe, PizzaGAN relies on image-level labels to perform stepwise generation of images mirroring cooking procedure. While interesting, the approach is tailored made for pizza image generation with limited kinds of ingredients and cooking methods being considered. Extension for dishes beyond pizza is not straightforward. The most relevant work to this paper is [21], which trains R^2GAN to generate 64 $\times$ 64 thumbnail images from recipes. Nevertheless, the design of R^2GAN is not from the angle of causality effect, but rather to learn cross-modal feature spaces for explainable recipe search. Specifically, R^2GAN treats ingredients and cooking steps independently and then collapses them into a feature for image generation and cross-modal learning. Different from this paper, the issue of how the cause of a cooking step results in a new effect for ingredients is not considered. To our best knowledge, the high-resolution food image generation considering causality is still an unexplored problem.

3. CookGAN

3.1. Model Architecture

The architecture of CookGAN is depicted in Figure 1. Given a recipe in text, we employ a recipe encoder, R^2GAN [21], to extract recipe features. R^2GAN embeds the features extracted from ingredients and cooking instructions respectively into a latent space compatible with food images. As demonstrated by R^2GAN, the embedded feature, denoted as φ_r, can be used to generate a thumbnail image of resolution 64 $\times$ 64. The basic idea of CookGAN is to progressively upsample the small-size image up to a resolution of 256 $\times$ 256. The key component of CookGAN is Cooking Simulator (Section 3.2), which generates cause-and-effect visual scenario.

CookGAN contains three pairs of generators and discriminators $\{(G_0, D_0), (G_1, D_1), (G_2, D_2)\}$. Initially, an embedded feature φ_r is concatenated with random noise z sampled from a Gaussian distribution $Z \sim \mathcal{N}(0, I)$. The result is input to an upsampling block, which is a multi-layer feedforward network that transforms the perturbed features into hidden image features V_0. The first generator G_0 pro-

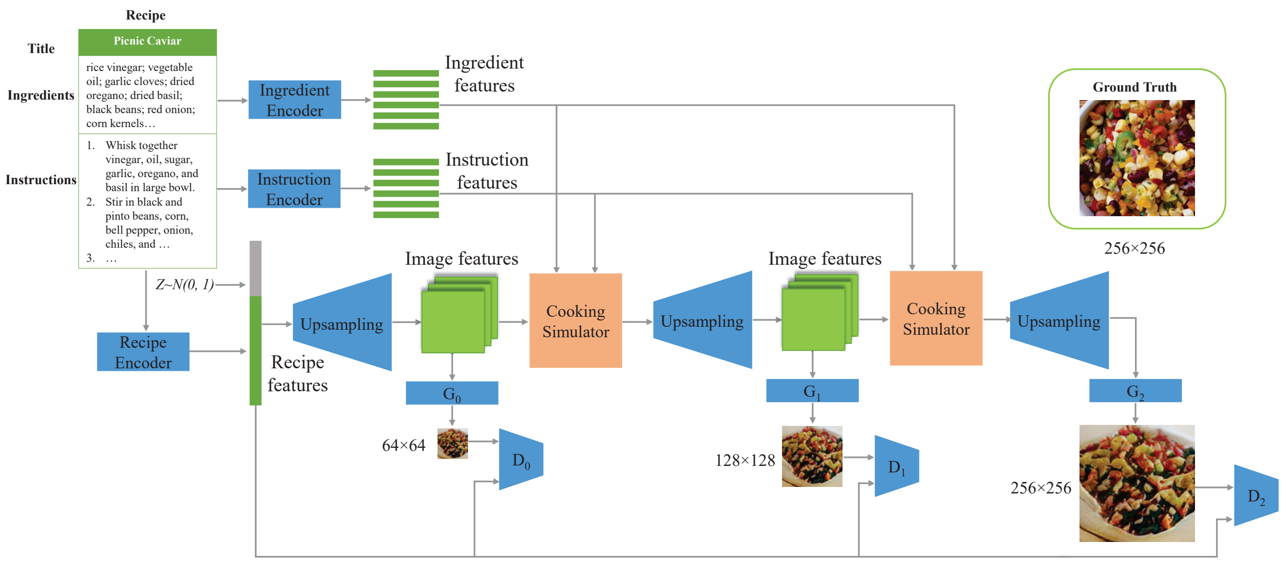

Recipe
Title
Picnic Caviar
Ingredients
rice vinegar; vegetable oil; garlic cloves; dried oregano; dried basil; black beans; red onion; corn kernels…
Instructions
1. Whisk together vinegar, oil, sugar, garlic, oregano, and basil in large bowl.
2. Stir in black and pinto beans, corn, bell pepper, onion, chiles, and …
3. …
Ingredient Encoder
Ingredient features
Instruction Encoder
Instruction features
Recipe Encoder
Z~N(0, 1)
Recipe features
Upsampling
Image features
Cooking Simulator
G0
G1
G2
D0
D1
D2
64×64
128×128
256×256
Ground Truth
256×256

Figure 1. The food image of a recipe is progressively upsampled from resolution of 64 × 64 to 256 × 256. The Cooking Simulator is a tailor-made sub-network to implicitly model cause-and-effect visual change in cooking process.

duces 64 × 64 images using V_0. The features V_0 are also fed into Cooking Simulator for the preparation of features in the next round of image upsampling. The process is repeated for two times, where G_1 and G_2 generate 128 × 128 and 256 × 256 images respectively.

CookGAN learns image distributions at different scales in an end-to-end fashion. Each generator is in charge of capturing distribution at a particular scale along with a discriminator to distinguish between real and fake images. Same as vanilla GAN [1], generator and discriminator are trained in an adversarial manner. Specifically, the three discriminators are trained separately, each of which used for training the corresponding generator. Meanwhile, the entire pipeline of image generation network at different scales, including the three upsampling blocks, generators and Cooking Simulators, are trained jointly. The objective function is defined as following:

$$L = \sum_{i=0}^{2} L_{G_i} + \lambda L_{CA}, \tag{1}$$

where L_{G_i} is the i_{th} generator loss and L_{CA} is conditioning augmentation [19] loss. The parameter λ is a trade-off hyperparameter to balance the two losses.

The generator loss consists of unconditional and conditional loss terms. The former loss is derived from discriminator in distinguishing between real and fake images. The latter term is to evaluate the degree-of-match between an generated image and its embedded feature initially extracted from R^2GAN. The i_{th} generator loss is defined as:

$$\begin{aligned} L_{G_i} =& \frac{1}{2}(\underbrace{\mathbb{E}_{\varphi_r \sim p_r, z \sim p_z}[\log(1 - D_i(G_i(\varphi_r, z))]}_{\text{unconditional loss}} + \\ & \underbrace{\mathbb{E}_{\varphi_r \sim p_r, z \sim p_z}[\log(1 - D_i(G_i(\varphi_r, z), \varphi_r)])}_{\text{conditional loss}}. \end{aligned} \tag{2}$$

The discriminator loss, similarly, contains two pairs of unconditional and conditional loss terms, as following:

$$\begin{aligned} L_{D_i} =& -\frac{1}{2}(\underbrace{\mathbb{E}_{x_i \sim p_{data_i}}[\log D_i(x_i)]}_{\text{unconditional loss}} + \\ & \underbrace{\mathbb{E}_{x_i \sim p_{data_i}, \varphi_r \sim p_r}[\log D_i(x_i, \varphi_r)]}_{\text{conditional loss}} + \\ & \underbrace{\mathbb{E}_{\varphi_r \sim p_r, z \sim p_z}[\log(1 - D_i(G_i(\varphi_r, z)))]}_{\text{unconditional loss}} + \\ & \underbrace{\mathbb{E}_{\varphi_r \sim p_r, z \sim p_z}[\log(1 - D_i(G_i(\varphi_r, z), \varphi_r))]}_{\text{conditional loss}}), \end{aligned} \tag{3}$$

where x_i is sampled from from the i_{th} scale of real food image distribution. Note that the i_{th} discriminator is only responsible for distinguishing between real images x_i and fake images $G_i(\varphi_r, z)$ in i_{th} scale.

Inspired by the effectiveness of conditioning augmentation in StackGAN++ [19], L_{CA} loss is employed as a regularizer to avoid overfitting and enforce smooth sampling from the recipe embedding manifold. To be specific, condition vectors are sampled from an independent Gaussian dis-

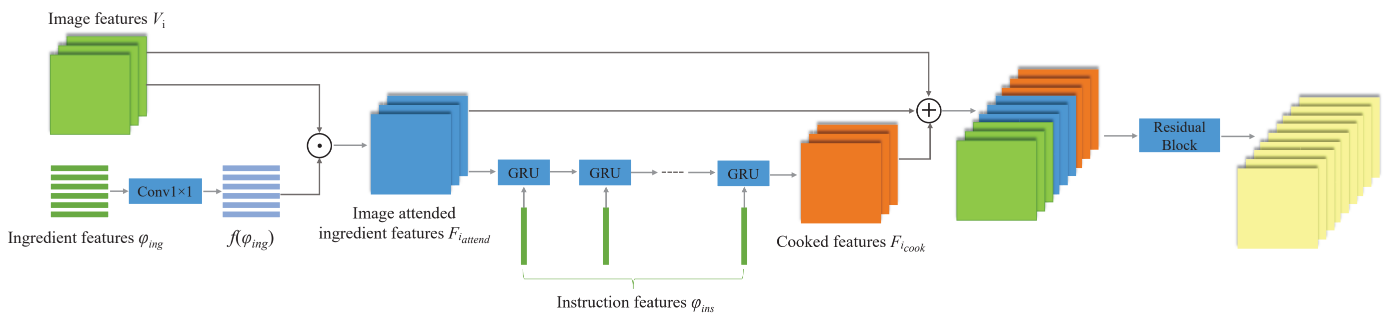


Image features V_i
Conv1×1
Ingredient features φ_{ing}
$f(\varphi_{ing})$
Image attended ingredient features $F_{i_{attend}}$
GRU
GRU
GRU
Instruction features φ_{ins}
Cooked features $F_{i_{cook}}$
Residual Block

Figure 2. Cooking Simulator prepares image attended ingredient features and "cooked" features for upsampling of food image. The GRUs are initialized with the image attended ingredient features. At each step of cooking, a GRU cell turns the image attended ingredient features into a new hidden state, modeling the change in ingredients as result of a cooking action.

tribution $\mathcal{N}(\mu(\varphi_r), \Sigma(\varphi_r))$, where mean $\mu(\varphi_r)$ and diagonal covariance matrix $\Sigma(\varphi_r)$ are parameterized by recipe features φ_r. L_{CA} is defined as the Kullback-Leibler divergence between $\mathcal{N}(\mu(\varphi_r), \Sigma(\varphi_r))$ and Gaussian distribution $\mathcal{N}(0, I)$:

$$L_{CA} = D_{KL}(\mathcal{N}(\mu(\varphi_r), \Sigma(\varphi_r))||\mathcal{N}(0, I)). \qquad (4)$$

3.2. Cooking Simulator

The intuition of Cooking Simulator is to imitate real cooking scenario, where different cutting and cooking actions are incrementally imposed on ingredients over time. Each action transforms some ingredients into a new form with change in composition, color or shape. For examples, "carrot" is cut into slice, "spaghetti" turns into black when stir fried with "squid sauce". The next subsequent actions can additively make change on this form along the cooking process.

Figure 2 depicts the network of Cooking Simulator. Denote $\varphi_{ing} = \{\varphi_{ing_m}\}_{m=1}^{M} \in \mathbb{R}^{M \times d_{ing}}$ as a list of ingredients, where φ_{ing_m} represents the d_{ing}-dimensional vector of m_{th} ingredient in the list. Furthermore, the image features of i_{th} scale are denoted as $V_i = \{v_j\}_{j=1}^{C} \in \mathbb{R}^{C \times L}$, where C is the channel depth and $L = W \times H$ is the resolution of a feature map. Initially, the image features V_i are fused with the list of ingredient features φ_{ing} to produce image attended ingredient feature maps, where the size of each map is $C \times L$. The j_{th} channel of $F_{i_{attend}}$ is computed by:

$$F_{i_{attend_j}} = \sum_{m=0}^{M-1} \sigma(v_j^T \cdot f(\varphi_{ing_m}))f(\varphi_{ing_m}), \qquad (5)$$

where $f(\cdot)$ is a 1×1 convolution to map ingredient features to the same dimension as the i_{th} scale hidden image features V_i. The operator $\sigma(\cdot)$ is softmax function that outputs an attention map of size L with probability values to indicate the spatial distribution of an ingredient. Through the softmax function, an attention map, i.e., $\sigma(v_j^T \cdot f(\varphi_{ing_m}))$, is generated for each ingredient. The spatial location of the ingredient is attended by multiplying the map with the corresponding ingredient feature $f(\varphi_{ing})$. Equation 5 performs weighted linear sum of the result for each ingredient to form the j^{th} channel image attended ingredient feature map.

Next, the cooking steps are sequentially encoded with Gated Recurrent Unit (GRU). The GRU cell is initialized with the image attended ingredient features $F_{i_{attend}}$, as shown in Figure 2. This design aims to imitate the process of cooking, where the ingredients are "cooked" by one step at a time. The result of cooking in a step, i.e., the hidden state of a GRU, is fed into the next GRU cell for subsequent cooking actions. Denote $\varphi_{ins} = \{\varphi_{ins_n}\}_{n=1}^{N} \in \mathbb{R}^{N \times d_{ins}}$ as a sequence of cooking steps, where φ_{ins_n} represents the d_{ins}-dimensional vector of n_{th} instruction in the sequence. The last hidden state of GRU for j_{th} channel of cooked feature map is formalized as:

$$F_{i_{cook_j}} = GRU(F_{i_{attend_j}}, \varphi_{ins}). \qquad (6)$$

where $F_{i_{cook_j}}$ denotes the j_{th} channel of "cooked" feature map at i_{th} scale. The final cooked feature maps share the same dimension as V_i with C number of channels and size equals to $W \times H$. To this end, the three groups of feature maps, V_i, $F_{i_{attend}}$ and $F_{i_{cook}}$, are concatenated and fed into residual blocks [3]. The transformed feature maps become the input for next round of image upsampling.

3.3. Implementation Details

The initial embedded feature extracted from R^2GAN [21] is in 1,024 dimensions, i.e., $\varphi_r \in \mathbb{R}^{1024}$. The ingredient encoder employs word2vec embedding to transform an ingredient from word into a high dimensional vector, i.e., $\varphi_{ing} \in \mathbb{R}^{300}$. The instructor encoder is based on skip-thoughts technique [8] as in [14], which encodes a step sentence into a fixed length vector. The resulting instruction feature is in 1,024 dimensions, i.e., $\varphi_{ins} \in \mathbb{R}^{1024}$. The channel depth of hidden image features is set to be 32 (Figure 1 and Figure 2). Inspired by PatchGAN [5, 22] and local adversarial image loss [20], each feature map in the image attended ingredient features (Figure 2) is divided into 32×32 patches. Cooking Simulator processes up to 10 cooking steps. Instructions beyond ten steps are truncated in consideration of the computational time.

Following [19, 16], Adam solver [7] with a learning rate of 0.0002 is adopted to train all the models. The balance factor in Equation 1 is set to be $\lambda = 1$. All the models are trained from scratch for 50 epochs.

4. Experiments

We validate CookGAN by visual quality assessmen-t (Section 4.2), semantic interpretation (Section 4.3) and content manipulatability (Section 4.4). Semantic interpretation includes the three tasks: (a) ingredient recognition on the CookGAN generated images, (b) retrieval of recipe using its generated image as query, and (c) retrieval of food image that is prepared with the same recipe as the generated query image. Content manipulatability is to test how a generated image reacts to change in cooking process, for example, by adding or removing ingredients in a recipe.

4.1. Experimental Settings

Dataset. The experiments are conducted on Recipe1M [14], which is the only publicly available large-scale dataset with paired recipes and images. The dataset provides 340,922 recipe-image pairs, with 70% for training, 15% for validation and 15% for tesing. As learning process of CookGAN for generating high resolution food images is slow and memory consuming, only validation set is used for model training. The R^2GAN [21], which extracts the initial stage of recipe features, is trained and validated on the training and validation sets respectively. The average number of instructions per recipe is 9 in original validation set.

Evaluation Metrics. Inception score (IS) is used to e-valuate the visual quality of generated images. As studied in [12], IS is correlated with human perception. A higher value of IS indicates better visual diversity and quality. We randomly sample 30,000 recipes from the test set for image generation. Comparison is made against state-of-the-art techniques including StackGAN++ [19] and R^2GAN [21]. The retrieval performance is based on median rank (MedR) among the true positives retrieved for testing queries. A lower value of MedR indicates better ability in retrieval. Following [21, 14], the retrieval dataset is formed by random sampling of 1,000 recipes from test set. All the 1,000 recipes take turn to generate images as testing queries.

4.2. Visual Quality Assessment

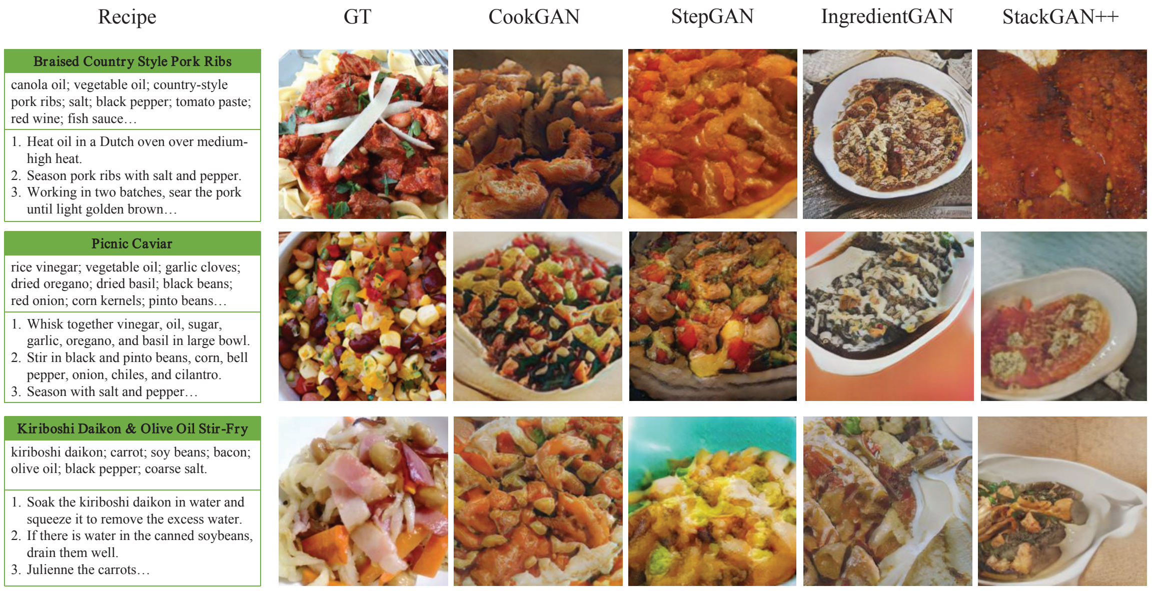
We compare CookGAN with StackGAN++ [19], the state-of-the-art text-to-image synthesis method, and R^2GAN [21], a baseline that generates low-resolution image. Ablation study is also conducted to compare with t-wo different versions of CookGAN with only ingredients (i.e., IngredientGAN) or cooking steps (i.e., StepGAN) being considered respectively. Note that, StepGAN still makes use of ingredient information. The major difference from CookGAN is that the input features to Residual blocks in Cooking Simulator do not involve image attended ingredient features ($F_{i_{attend}}$), i.e., only image features (V_i) and cooked features ($F_{i_{cook}}$) are used to produce images for next round. IngredientGAN, on the other hand, skips $F_{i_{cook}}$ and leverages only V_i and $F_{i_{attend}}$.

The result is listed in the 2_{nd} column of Table 1. When comparing to the baseline, which linearly interpolates the 64×64 thumbnails generated by R^2GAN [21] to the resolution of 256×256, all other approaches show consistently higher inception score (IS). Among them, CookGAN significantly outperforms others, including StackGAN++, in a large margin. CookGAN is more capable in modeling the color and texture distribution of a dish when composing d-ifferent ingredients. With reference to Figure 3, the images simulated by CookGAN show similar pattern of ingredien-t composition as the real sample images. Using the recipe "Picnic Caviar" as an example, the image is correctly presented with "porn kernels" in yellow and "pinto beans" in brown color. Furthermore, the simulated image vividly imitates the fuzzy composition of various ingredients in small size, resulting in a visually and structurally similar image as the original image. Although StackGAN++ manages to present the colors of some ingredients in red and yellow, the failure in simulating the original shapes and sizes of ingredients makes the generated image appear unnatural. In general, StackGAN++ is incapable of handling the recipes with relatively larger number of ingredients. The simulated images by StackGAN++ are often composed of only a few major ingredients. Thanks to image attended ingredien-t features, CookGAN is able to visually capture both major and supplementary ingredients. Further with Cooking Simulator to procedurally encode instructions, these ingredients are sometimes composed in way similar to food preparation in real world scenario.

As shown in Table 1, the IS scores drop dramatically when leveraging only ingredients or cooking steps. The ablation studies indicate that both ingredients and cooking instructions play a significant role in food image generation. On the one hand, when comparing to the case when cooking steps are not considered, IngredientGAN shows much lower IS score, drops by 11.5% compared with CookGAN. As shown in Figure 3, despite that IngredientGAN manages to generate images with relatively fine-grained texture pattern than StackGAN++, the dish colors are monotonic and lack of visual diversity, resulting in lower IS score. The monotone in color appearance is due to the fact that, in the

Method	Inception score↑	MedR (I2R)↓	MedR (I2I)↓
R^2GAN [21]	$4.54 \pm .07$	500.0	476.0
StackGAN++ [19]	$5.03 \pm .09$	144.5	147.0
IngredientGAN	$4.79 \pm .08$	84.5	115.5
StepGAN	$5.30 \pm .09$	77.0	123.0
CookGAN	$\mathbf{5.41 \pm .11}$	**64.0**	**108.0**

Table 1. Visual quality and semantics comparison in terms of inception score and median rank (MedR) respectively. "↑" indicates higher is better and "↓" indicates lower is better.



Braised Country Style Pork Ribs

canola oil; vegetable oil; country-style pork ribs; salt; black pepper; tomato paste; red wine; fish sauce…

1. Heat oil in a Dutch oven over medium-high heat.
2. Season pork ribs with salt and pepper.
3. Working in two batches, sear the pork until light golden brown…

Picnic Caviar

rice vinegar; vegetable oil; garlic cloves; dried oregano; dried basil; black beans; red onion; corn kernels; pinto beans…

1. Whisk together vinegar, oil, sugar, garlic, oregano, and basil in large bowl.
2. Stir in black and pinto beans, corn, bell pepper, onion, chiles, and cilantro.
3. Season with salt and pepper…

Kiriboshi Daikon & Olive Oil Stir-Fry

kiriboshi daikon; carrot; soy beans; bacon; olive oil; black pepper; coarse salt.

1. Soak the kiriboshi daikon in water and squeeze it to remove the excess water.
2. If there is water in the canned soybeans, drain them well.
3. Julienne the carrots…

Figure 3. Comparison of food images generated by CookGAN, StepGAN, IngredientGAN and StackGAN++.

design of CookGAN, the ingredients are mainly to prioritize image regions of different important levels for upsampling. Without Cooking Simulator that changes the appearances of ingredients in stepwise manner, learning photo-realistic image is ineffective.

Comparing to StepGAN, CookGAN boosts IS to 5.41 from 5.30 achieved by StepGAN. The result verifies the advantage of attending ingredient to regions for sequential upsampling of a small-size image to higher resolution versions. Based on the results, the image attended ingredient features indeed help to prevent ingredients in small size and quantity, which are usually supplementary ingredients, from being ignored during image upsampling. As shown in Figure 3, the images generated by StepGAN cannot capture the rich color and texture variations as in CookGAN.

4.3. Semantic Interpretation

We argue that a generated image should not only be visually appealing, but also semantically explainable. We design three tasks to measure the interpretability of generated images.

Task 1: Ingredient recognition is to multi-label the ingredients in an food image. We employ the ingredient decoder [13] pre-trained on Recipe1M [14] for this task. Comparison is made between the images generated by CookGAN and the real images prepared by their corresponding recipes. Using Intersection of Union (IoU) between the predicted and ground-truth labels as a measure, both real and synthetic images show almost the same performance of IoU = 0.29. The result basically verifies that the CookGAN synthesized images are as interpretable as the real images. Figure 4 lists the recognized ingredients of two sample images generated by CookGAN. Not only visible ingredients, but non-visible ingredients, such as "butter" for "Bajan sweet bread" and "pepper" for "Spanish pisto", can be recognized. The result is similar to that of original image. Limited by the accuracy of decoder, ingredients such as "chicken" and "potato" cannot be distinguished for both real and synthetic images. Similarly, "egg" and "baking soda", which commonly appear in recipes of cookies, are falsely detected for both kinds of images.

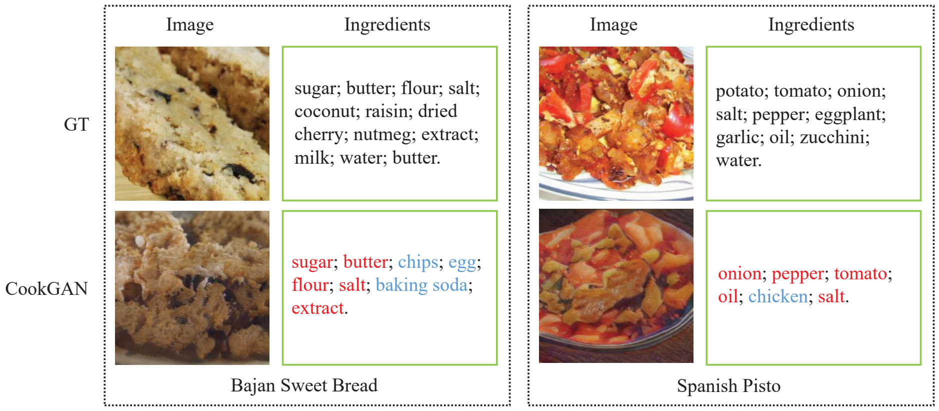


Image
Ingredients
Image
Ingredients
GT
sugar; butter; flour; salt; coconut; raisin; dried cherry; nutmeg; extract; milk; water; butter.
potato; tomato; onion; salt; pepper; eggplant; garlic; oil; zucchini; water.
CookGAN
sugar; butter; chips; egg; flour; salt; baking soda; extract.
onion; pepper; tomato; oil; chicken; salt.
Bajan Sweet Bread
Spanish Pisto

Figure 4. Ingredient recognition results of the images generated by CookGAN. Ingredients appeared in ground truth are highlighted in red and blue otherwise.

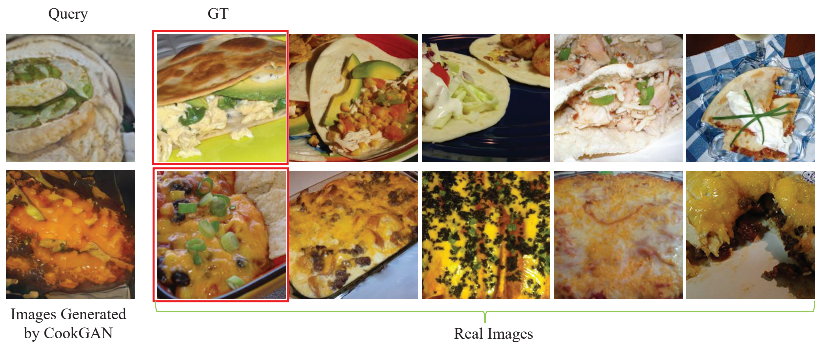


Figure 5. Examples of using food images generated by CookGAN to retrieve real images. Ground truth images are marked with red bounding box.

Task 2: Image-to-recipe retrieval (I2R). This is a reverse engineering task, where a generated image is treated as a query to retrieve its recipe from a dataset composed of 1K recipe-image pairs. We employ R^2GAN to extract the cross-modal features of images and recipes. Recipes are subsequently ranked in descending order of their similarities to a query image based on cosine similarity. The third columns of Table 1 shows the performances using images generated by different GANs. When directly using the thumbnails generated by R^2GAN for retrieval, the performance is close to random ranking of recipes. In contrast, by learning to progressively upsample from low-resolution to high-resolution, all other GANs significantly boost the performance.

Interestingly, the MedR performance is not necessarily correlated to visual quality based on IS. While IngredientGAN is lower in IS score than StackGAN++, its MedR is better by 60 ranks. The result clearly indicates the superior capability of CookGAN in encoding the semantics of food than StackGAN++. StepGAN shows slight improvement than IngredientGAN for being capable of disambiguating recipes with similar ingredients but different cutting and cooking methods. CookGAN, afterall, shows the best performance among all the approaches. Nevertheless, compared to using the original images as queries where the MedR can be as high as 1.0 [15], there is still a performance gap between using real and synthetic images for retrieval.

Task 3: Image-to-image retrieval (I2I). This task is to retrieve the real food images using the generated images. The result is listed in the 4_{th} column of Table 1. Slightly different trends of performance than I2R is observed. Specifically, the performance of IngredientGAN is better than StepGAN. We believe that this is due to the composition of dataset, where most of the recipes are only associated to one food image. As a consequence, when StepGAN generates an image with visually different ingredient composition from the sample image in the dataset, their similarity might be decreased. IngredientGAN, which merely retrieves similar images based on ingredient content, bypasses the issue due to fuzzy variations in ingredient composition

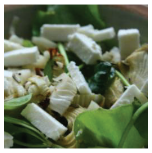
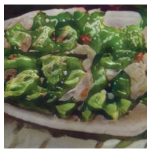
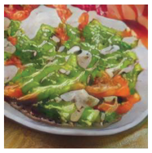
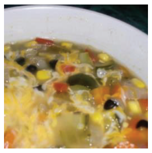
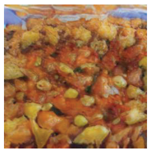
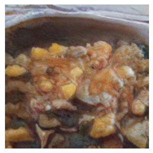
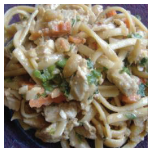
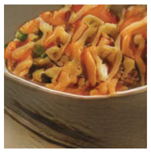
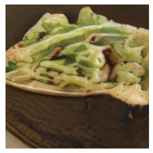
Recipe Manipulation	Concrete Operation	GT	Before Manipulation	After Manipulation
Add Ingredients	+ Carrot			
Minus Ingredients	- Carrot			
Replace Ingredients	Carrot$\rightarrow$ lettuce			

Figure 6. Examples of generated images by CookGAN with different recipe manipulation operators.

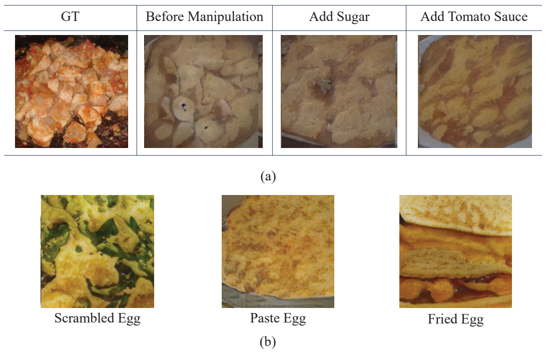


Figure 7. Examples showing manipulability of CookGAN in dealing with (a) visibility of ingredient, (b) different cooking actions on an ingredient.

and hence achieves lower MedR. Figure 5 shows the top-5 retrieved images by the queries generated by CookGAN. The retrieved images are not only visually but also semantically similar.

4.4. Content Manipulability

An advantage of CookGAN is that an image can be produced on-the-fly by incremental manipulation of a recipe, for example, through semantically changing ingredient list. In this section, we contrast the visual presentation of food images when they are prepared with slightly different ingredients from an original recipe. Figure 6 shows the examples of dishes when "carrot" is added, removed or replaced by "lettuce". In the case of addition, it can be seen that carrots (or items in orange color) are distributed throughout the dish. On the other hand, when "carrot" is removed, the dish becomes much less in orange color. In the case of replacement, the spaghetti is changed from orange to creamy white, and is covered with lettuce-like objects. More impressively, CookGAN manages to learn the visibility of ingredients in a dish. As shown in Figure 7 (a), the appearance of dish remains the same when "sugar" is added. On the contrary, the color changes when "tomato sauce" is added. Figure 7 (b) shows examples to demonstrate the bundle effect of ingredient + action. The appearance of "egg" varies dramatically depending on cooking method.

5. Conclusion

We have presented CookGAN for visual modeling of causality effect. Empirical results show that CookGAN is capable of synthesizing realistic visual scenario in depicting the cause-and-effect of cooking action. Comparing to StackGAN++, CookGAN manages to simulate proper color, shape and composition of ingredients due to cooking actions. Furthermore, CookGAN demonstrates commonsense response to manipulation operators, including dealing with transparency effect of ingredients and ingredient-action bundle effect. Compared to real images, similar performance is also reported when recognizing ingredients of synthetic images. While encouraging, CookGAN does not consider ingredient quantity and cooking style (e.g., home cooked style, sweet-and-sour), which will be our future work.

Acknowledgement

The work described in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (CityU 11203517).

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[2] Fangda Han, Ricardo Guerrero, and Vladimir Pavlovic. The art of food: Meal image synthesis from ingredients. *arXiv preprint arXiv:1905.13149*, 2019.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Daichi Horita, Wataru Shimoda, and Keiji Yanai. Unseen food creation by mixing existing food images with conditional stylegan. In *Proceedings of the 5th International Workshop on Multimedia Assisted Dietary Management*, pages 19–24. ACM, 2019.

[5] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[6] Yoshifumi Ito, Wataru Shimoda, and Keiji Yanai. Food image generation using a large amount of food images with conditional gan: ramengan and recipegan. In *Proceedings of the Joint Workshop on Multimedia for Cooking and Eating Activities and Multimedia Assisted Dietary Management*, pages 71–74. ACM, 2018.

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[9] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[10] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8002–8011, 2019.

[11] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *International Conference on Machine Learning*, pages 1060–1069, 2016.

[12] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[13] Amaia Salvador, Michal Drozdzal, Xavier Giro-i Nieto, and Adriana Romero. Inverse cooking: Recipe generation from food images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10453–10462, 2019.

[14] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3020–3028, 2017.

[15] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11572–11581, 2019.

[16] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1316–1324, 2018.

[17] Guojun Yin, Bin Liu, Lu Sheng, Nenghai Yu, Xiaogang Wang, and Jing Shao. Semantics disentangling for text-to-image generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2327–2336, 2019.

[18] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5907–5915, 2017.

[19] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1947–1962, 2018.

[20] Zizhao Zhang, Yuanpu Xie, and Lin Yang. Photographic text-to-image synthesis with a hierarchically-nested adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6199–6208, 2018.

[21] Bin Zhu, Chong-Wah Ngo, Jingjing Chen, and Yanbin Hao. R2GAN: Cross-modal recipe retrieval with generative adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11477–11486, 2019.

[22] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[23] Minfeng Zhu, Pingbo Pan, Wei Chen, and Yi Yang. Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5802–5810, 2019.