How to make a pizza: Learning a compositional layer-based GAN model

MIT, CVPR 2019

Making a pizza

- Pizza recipe
  - Dough: Strong flour, salt, yeast, olive oil, sugar, water (dark beer)
  - Topping
    - Tomato source
    - Pineapple
    - Cheese (mozzarella, cheddar, …)
    - Potato, sweet potato
    - Green pepper, Olive, mushroom
    - Bacon, Pepperoni, shrimp
  - Baking
    - 180℃, 20 minutes

  - Do we put toppings all together? No!

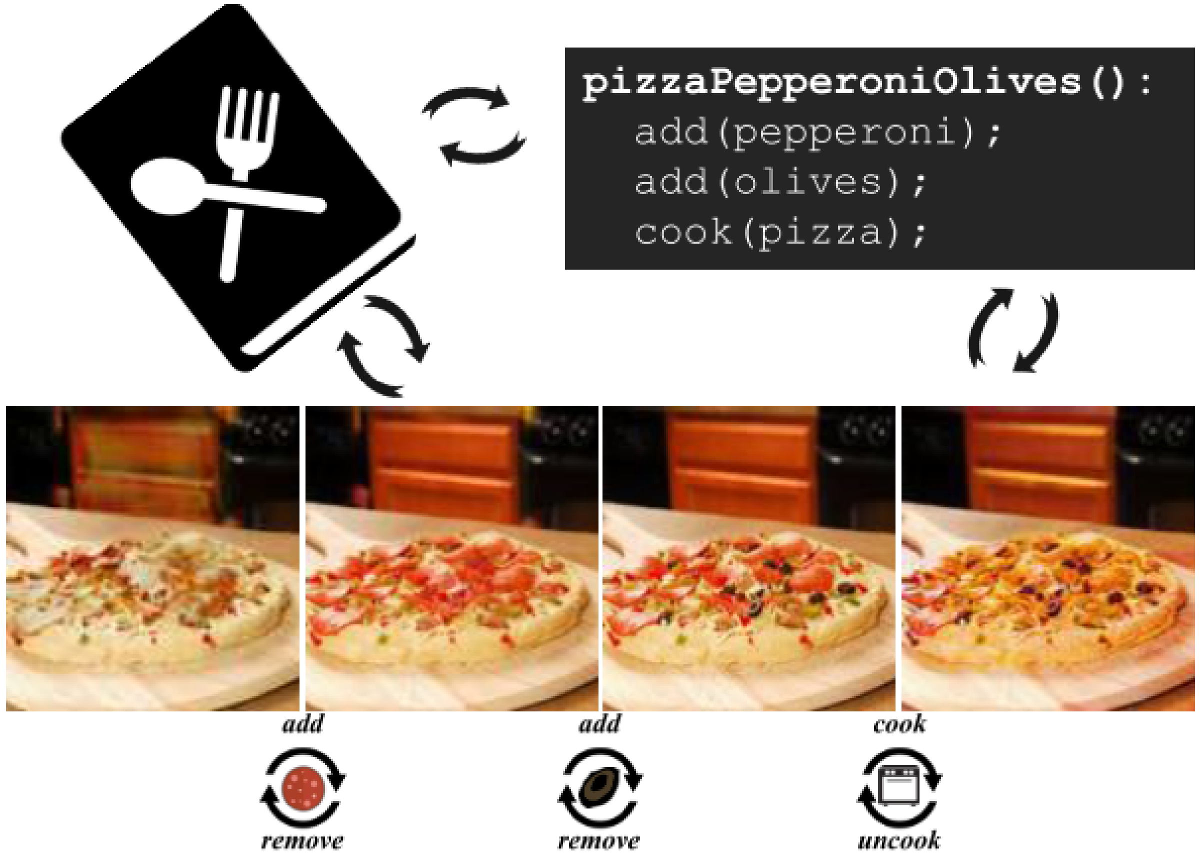
Step-by-step procedure

- Making pizza with traditional GANs:
    - topping_class = [dough, tomato, peperoni, pineapple, olive, cheese, …]
    - pepperoni_pizza = [1, 1, 1, 0, 0, 1, …]
    - pineapple_pizza = [1, 1, 0, 1, 0, 1, …]

- A recipe is an ordered set of instructions (e.g., put toppings)
    - Dough → tomato → peperoni → cheese
    - Dough → tomato → pineapple → potato → bacon → cheese
    - …

- We need to teach “step-by-step procedure” to GANs

PizzaGAN: Overview



```
pizzaPepperoniOlives():
  add(pepperoni);
  add(olives);
  cook(pizza);
```

Figure 1. **How to make a pizza:** We propose *PizzaGAN*, a compositional layer-based generative model that aims to mirror the step-by-step procedure of making a pizza.

PizzaGAN: Overview

- Creating the pizza image requires sequentially rendering different ingredient layers on top of a pizza dough image
    - In this viewpoint, we can consider the reverse order representation

- Layering
    - Key concept of digital image drawing and editing
    - e.g., Photoshop, CorelDraw, GIMP, …
    - PizzaGAN is a compositional layer-based generative model

- Related topic
    - Image translation, image inpainting, modularity, …

PizzaGAN: Problem setup

- Dataset
  - Image $I \in \mathbb{R}^{H \times W \times 3}$
  - Image-level label $C = \{c_1, c_2, \ldots, c_k\}$ be the set of all k different toppings (e.g., pepperoni, pineapple, ⋯)
  - Dataset is split into two domains: one with the images which contain the class c (X_c^+) and one with the images that do not contain it (X_c^-)

- Goal
  - For each class c, learn two mapping functions that translate images without instance of class c to images with instance of class c and *vice versa*
    (i.e., adding / removing topping c)

PizzaGAN: Method

r: real
f: fake

- Generator module
  - G_c^+: the generator module that add a layer of the class c on image I^{r-} (mapping $G_c^+ : X_c^- \to X_c^+$)
  - G_c^-: ⋯ remove ⋯ (mapping $G_c^- : X_c^+ \to X_c^-$)
  - Output generated images $I^{f+} = G^+(I^{r-})$ and $I^{f-} = G^-(I^{r+})$ are given by:

$$I^{f+} = M^+ \odot A^+ + (1 - M^+) \odot I^{r-}$$
$$I^{f-} = M^- \odot A^- + (1 - M^-) \odot I^{r+}$$

where $M^+, M^- \in [0,1]^{H \times W}$ are masks that indicate how each pixel of adding or removing layer,
$A^+ \in \mathbb{R}^{H \times W \times 3}$ is the RGB image that captures the appearance of the adding layer, while A^- captures that lies underneath the added layer

PizzaGAN: Method

- Generator module
  - Based on CycleGAN architecture
  - A convolutional layer is added to predict masks

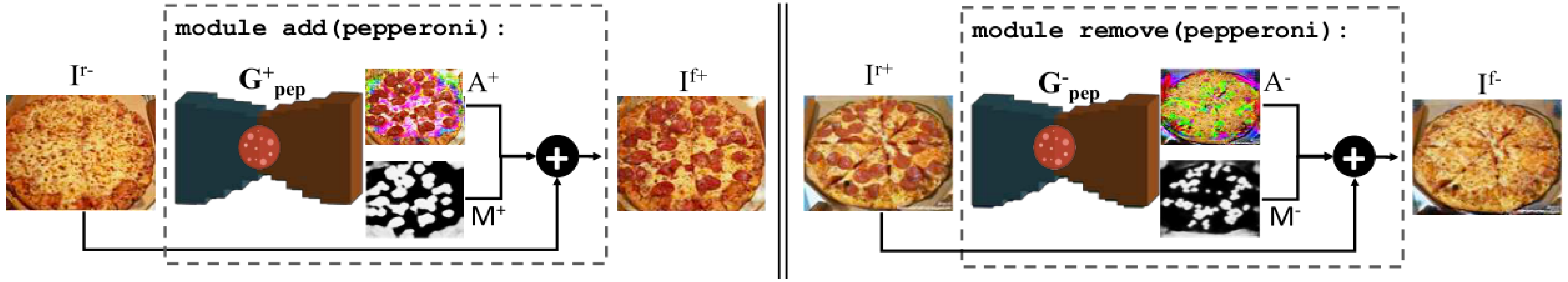


Figure 2. **Module operators that are trained to add and remove pepperoni on a given image.** Each operator is a GAN that generates the appearance A and the mask M of the adding or the removing layer. The generated composite image is synthesized by combining the input image with the generated residual image.

PizzaGAN: Method

- Discriminator
  - PizzaGAN contains a single discriminator D
    → distinguish whether the input image is real of fake
    →perform a multi-label classification task for all classes (toppings)
  - Based on PatchGAN with modification for multi-label classification

- Training
  - All generator modules and the discriminator are learned jointly
  - Four losses
    - Adversarial loss
    - Classification
    - Cycle consistency: Training images are not paired
    - Mask regularization: To prevent masks converge to zero (masks to be one)

PizzaGAN: Method

- Test time inference
  1. The input image are fed into the discriminator, and toppings appear in the image are classified
  2. The order of toppings (from the bottom to the top) is computed by overlapping masks
  3. Sequentially remove existing toppings and add new toppings

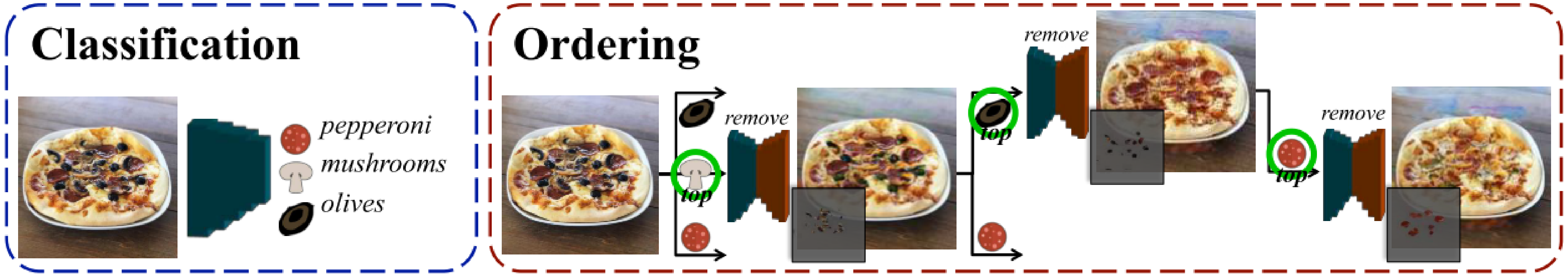


Figure 3. **Test time inference.** Given a test image, our proposed model detects first the toppings appearing in the pizza (classification). Then, we predict the depth order of the toppings as they appear in the input image from top to bottom (ordering). The green circles in the image highlight the predicted top ingredient to remove. Using this ordering, we apply the corresponding modules sequentially in order to reconstruct backwards the step-by-step procedure for making the input pizza.

PizzaGAN: Method

- Test time inference

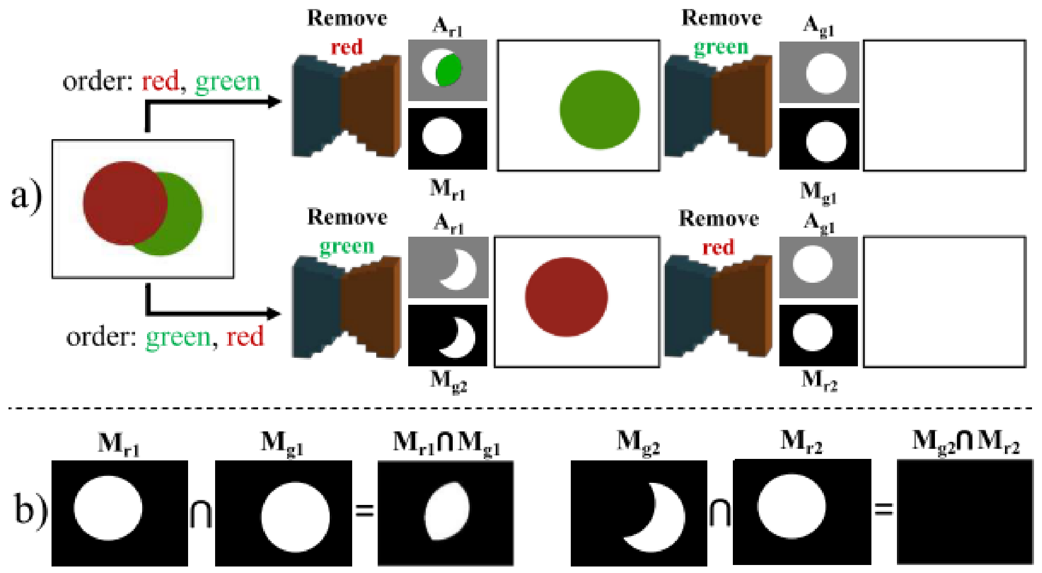


Figure 4. **Predicting the ordering of layers**. (a) A toy example with two overlapping circular objects (red on top of green). In the first row we first remove the red object and then the green one, while in the second row we follow the reverse order. (b) Intersection between the two generated masks for each ordering permutation. We observe that in the first case the two generated masks M highly overlap, while in the second one the overlap is zero.

Pizza dataset

- For the experiment, the authors collected “real pizzas” and created “synthetic pizzas
  - Real pizza images are collected from Instagram (#pizza) and annotated by using AMT
  - Challenges in annotation
    - Visually similar toppings (e.g., bacon-ham, basil-spinach)
    - Majority vote by five different annotators

  - The real pizza set
    - 9,213 annotated pizza images
    - The average number of toppings is 2.9 with a *std* of 1.1
  - The synthetic pizza set
    - 5,500 pizza images up to ten toppings
    - bacon, basil, broccoli, mushroom, olive, onion, pepperoni, pepper, pineapple, tomato

Pizza dataset

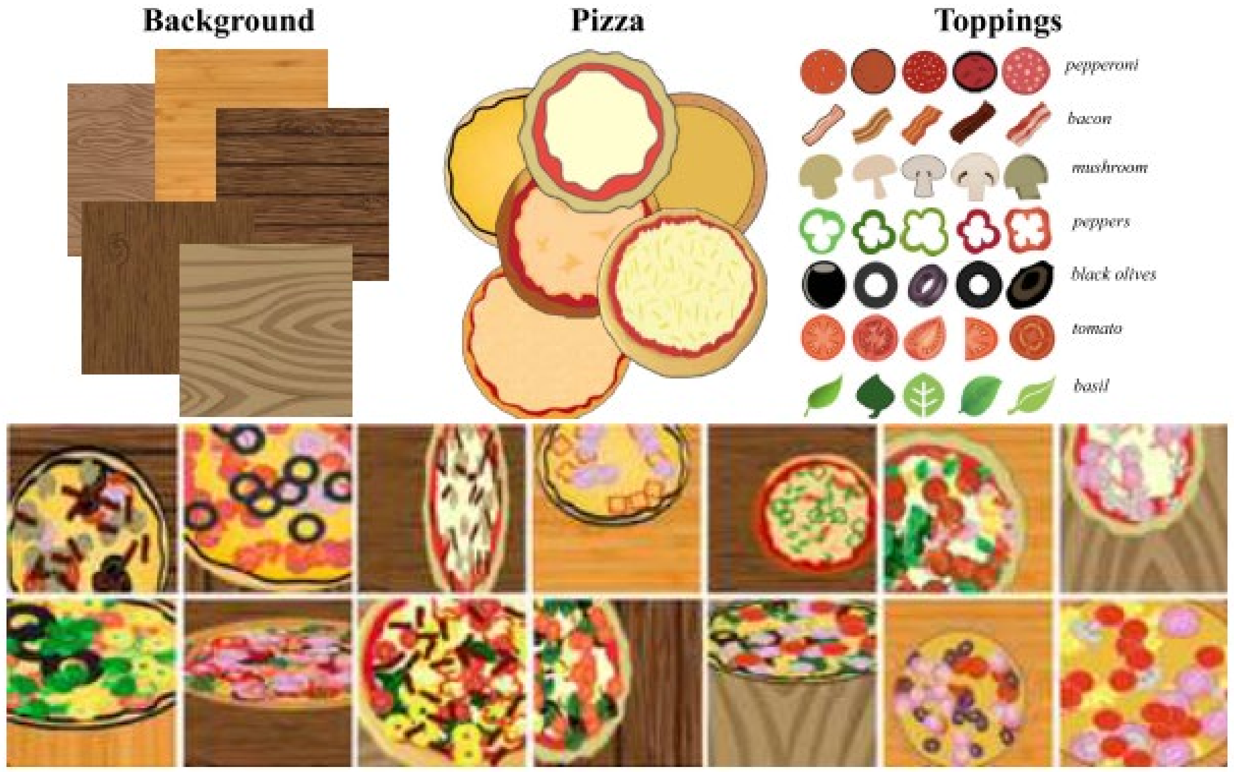


Figure 6. **Creating synthetic pizzas.** Top: Examples of background textures, base pizza images, and toppings used to create synthetic pizzas. Bottom: Examples of created synthetic pizzas.

Experimental result

- Removing toppings from pizza

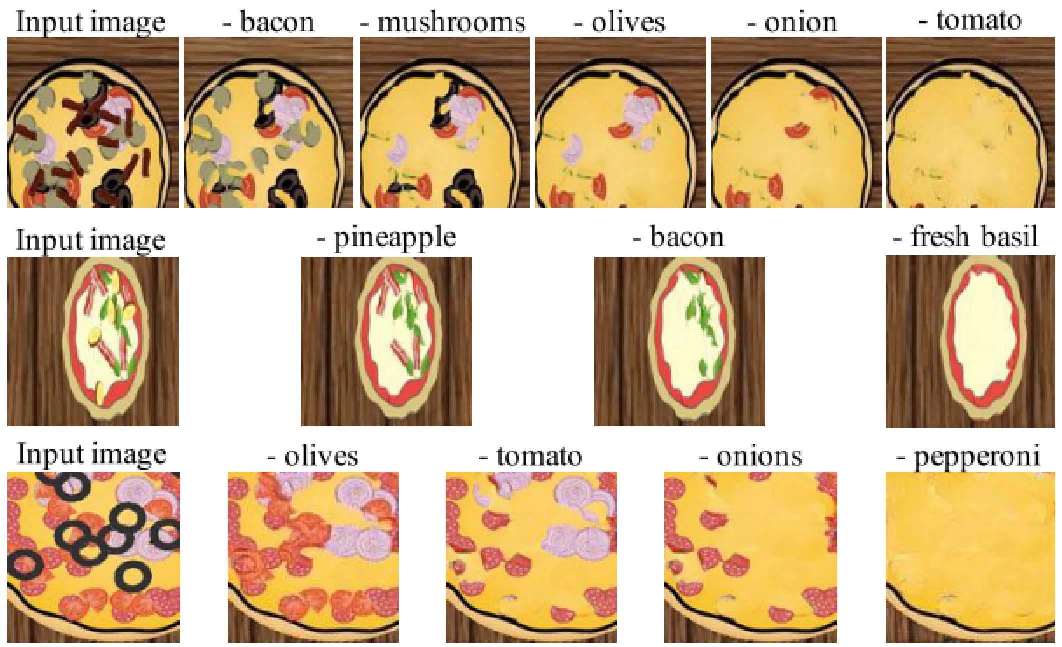


Figure 8. We predict the sequence of removing operators and apply them sequentially to the input image. Note how every time the current top ingredient is the one removed. This process reveals several invisible parts of ingredients when removing the top layers that occlude them.

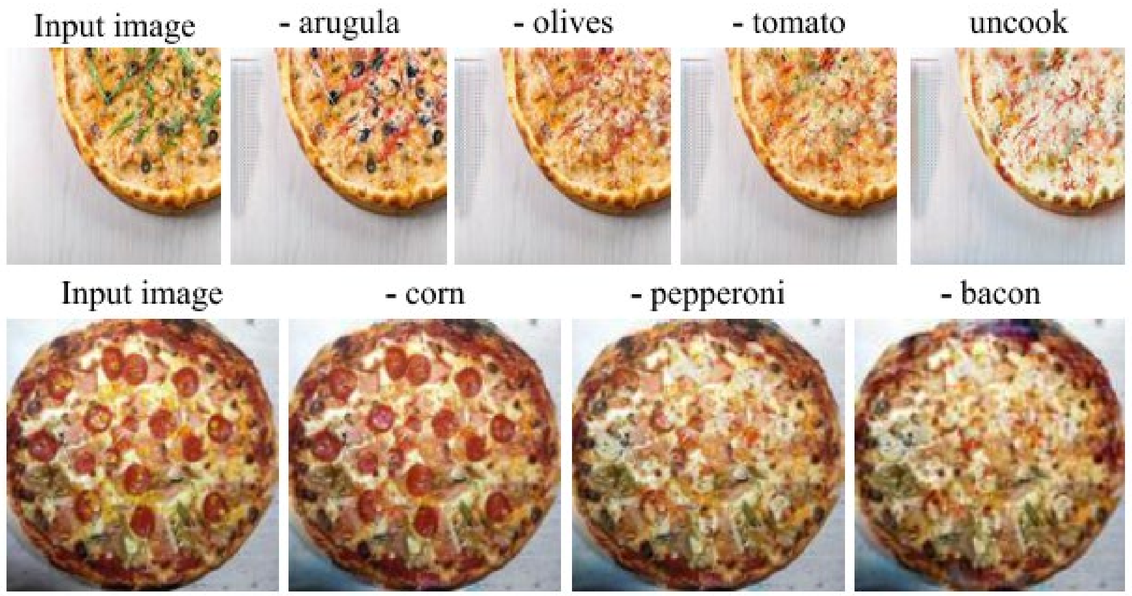
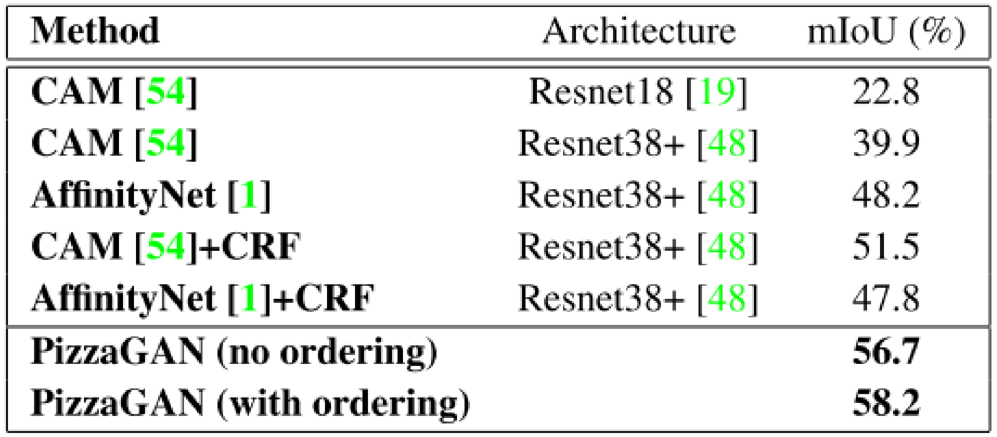


Figure 10. **PizzaGAN**: We predict the sequence of the operaors and apply them sequentially to the input image. The goal of the model is to remove every time the top ingredient. This leads in reconstructing backwards the recipe procedure used to make the input pizza.

Experimental result

- Classification of toppings and mask prediction (segmentation)
  - Classification acc. is near 99%



Method	Architecture	mIoU (%)
CAM [54]	Resnet18 [19]	22.8
CAM [54]	Resnet38+ [48]	39.9
AffinityNet [1]	Resnet38+ [48]	48.2
CAM [54]+CRF	Resnet38+ [48]	51.5
AffinityNet [1]+CRF	Resnet38+ [48]	47.8
PizzaGAN (no ordering)		**56.7**
PizzaGAN (with ordering)		**58.2**

Table 1. **Weakly-supervised segmentation mIoU performance on synthetic pizzas.** All comparison methods are pre-trained on ILSVRC, while PizzaGAN is trained from scratch. In Resnet38+, GAP and FC layers replaced by three atrous convolutions [7].

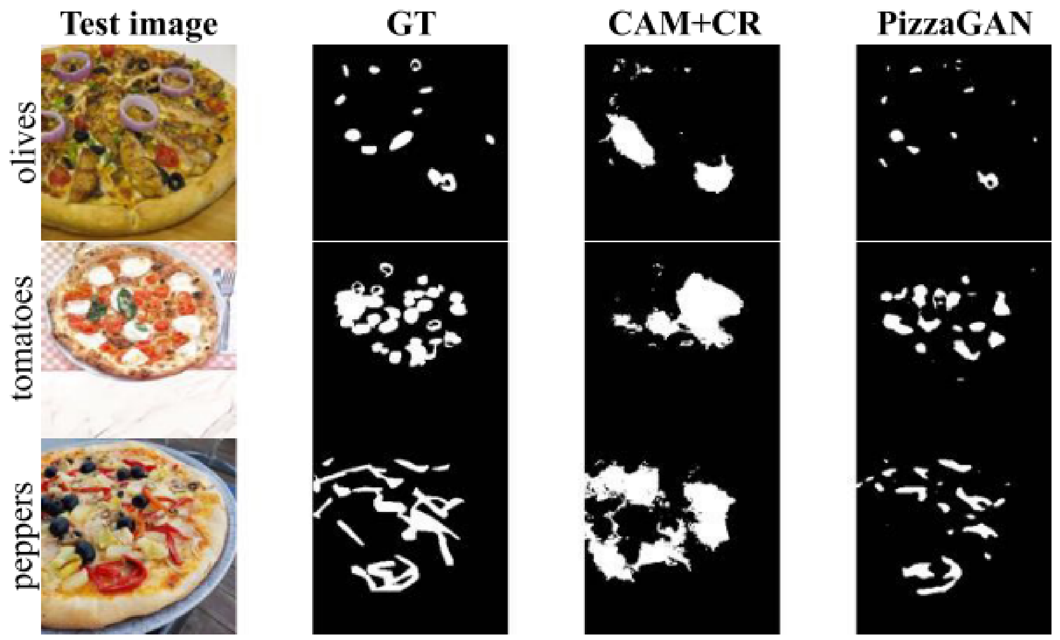


Figure 11. Examples of segmentation results on real pizzas.

Experimental result

- Ordering
  - Metric: Damerau-Levenshtein distance (DL) → edit distance
    - DL considers “deletion”, “insertion”, “substitution”, “transposition”

$$d_{a,b}(i,j) = \min \begin{cases} 0 & \text{if } i = j = 0, \\ d_{a,b}(i-1,j)+1 & \text{if } i > 0, \\ d_{a,b}(i,j-1)+1 & \text{if } j > 0, \\ d_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} & \text{if } i,j > 0, \\ d_{a,b}(i-2,j-2)+1 & \text{if } i,j > 1 \text{ and } a_i = b_{j-1} \text{ and } a_{i-1} = b_j, \end{cases}$$

  - Result (lower is better)
    - PizzaGAN: 0.33
    - Random sequence of the oracle labels: 0.42
    - Random sequence of a random labels: 0.91

Conclusion

- Layer decomposition problem can be solved as sequential un-paired image-to-image translation
  - Classifying, segmenting, and ordering the layers can be done in a weakly-supervised manner without any pixel-, depth-, and order-level supervision

- Beyond pizza, concept of layering could be applied to
  - Digital image drawing and editing
  - Virtual try-on
  - …