CS230

Conditional Generative Adversarial Models for Generating Images of Food

Eric Wu
wueric@stanford.edu

Sudip Guha
sudipg@stanford.edu

Abstract

We present a conditional generative adversarial network for generating class conditioned food images. We train several model architectures, including vanilla cWGAN-GP, ResNet cWGAN-GP, and AC-WGAN-GP, on the Food-101 dataset to generate images of food conditioned on the category. We then evaluate the quality, diversity, and conditioning of these models by calculating Inception score and by using a classifier to classify the generated images into their intended categories.

1 Introduction

Using conditional generative adversarial networks (conditional GANs) to generate images given a conditional input is currently an active area of research. Conditional GANs consist of two neural networks, a generator network that synthesizes a fake image given a conditional input, and a discriminator network that determines whether or not an image was generated by the generator network, trained in an adversarial manner (1). Generating images of food using GANs is a particularly challenging problem, due to the highly varied fine geometric structure of food. In this paper, we train several conditional GAN models using the Food-101 dataset to generate images of food based on category using a one-hot vector of food category labels as the condition, and evaluate the models based on the quality and diversity of the images produced as well as the efficacy of the conditioning.

2 Related work

There have been several previous efforts to generate images of food given some condition. Using the much larger Recipe1M dataset, Bar El *et al.* trained a conditional StackGAN++ to generate images of food conditioned on text annotation (2). The images generated by this model tended to look pleasing to the eye and appeared qualitatively to look like food. By human inspection, the model generally performed well for images of foods with uniform consistency (i.e. porridge), but performed poorly for images of food with well-defined geometric structure (i.e. burgers). Ito *et al.* (3) trained cGAN and cWGAN-GP models to generate conditional images of ramen conditioned on categorical one-hot vectors, and images of food conditioned on text embeddings. Like the Bar El *et al.* paper, the images generated by these models looked like food but lacked most of the detailed geometric structure expected in a real image.

3 Dataset and Preprocessing

We used the Food-101 dataset provided by ETH Zurich (4). Food-101 consists of 101,000 images of food, with 1,000 images each for 101 different categories of food. Food-101 contains mislabeled images, and images within the same class have substantial variance in terms of subject matter, lighting conditions, and background. Example images are shown in Figure 1.

CS230: Deep Learning, Spring 2019, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

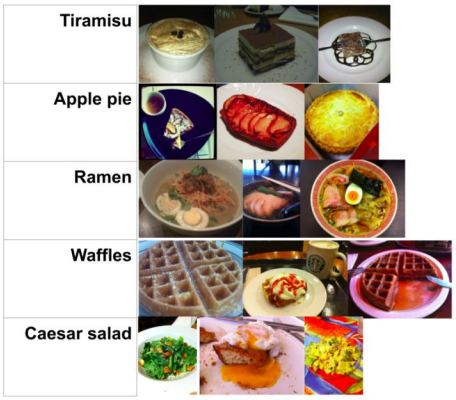

Figure 1: Example images drawn from a few categories of the Food-101 dataset. Note the substantial variation in the images in subject matter, lighting conditions, and background, and the mislabeled image (the second image for Caesar salad).

We used a subset of 10 classes to train our models to reduce the difficulty of conditional image generation. These 10 classes were: (1) Caesar salad; (2) tiramisu; (3) clam chowder; (4) pancakes; (5) guacamole; (6) pad thai; (7) ramen; (8) spaghetti bolognese; (9) waffles; and (10) apple pie. These classes were chosen for their variety of structure and color.

3.1 Data Preprocessing and Augmentation

We preprocessed the images in the following way: (1) Crop the image into the largest possible square located at the center; (2) Resample and resize the image to 64x64; (3) Rescale the pixel values of the image from [0, 255] to [-1, 1].

We augmented the images at training time by applying random left/right flips, random color modifications, and random crops followed by resampling back to 64x64.

4 Methods

4.1 Models

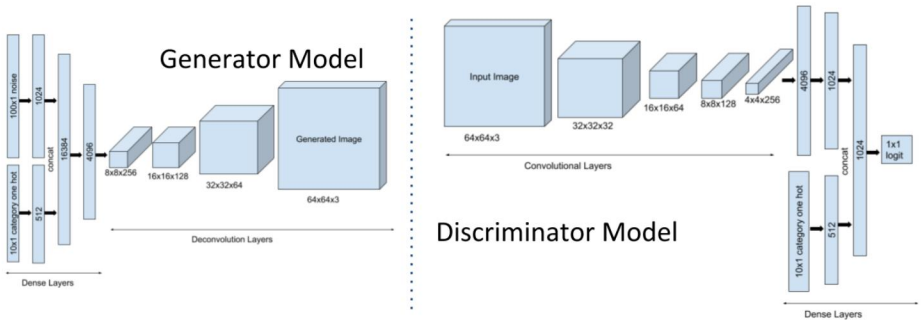

Figure 2: Simplified block diagrams of our generator and discriminator models.

Our model is a GAN with a generator and discriminator. The inputs to the generator are a 100-dimensional noise vector z and a 10-dimensional one-hot label vector y identifying the class of the real image x. The generator outputs a 64x64x3 image $\widetilde{x}$ which is bounded to the (-1, 1) range because the final layer of the generator has a $\tanh$ activation.

The Discriminator network takes either x or $\widetilde{x}$ and the one-hot class label y. The output of the discriminator in the cWGAN-GP and ResNet cWGAN-GP case was a single logit $D(*)$ corresponding to whether the image is real or generated. Meanwhile, in the AC-WGAN-GP case, the discriminator determines both whether the image was real or generated, and what class the image is, and hence outputs both a single logit $D(*)$ as well as another ten-dimensional logit corresponding to the class of the image.

Batch normalization in the generator improved training speed (higher learning rate and higher quality images). However, we did notice that using batch normalization had the effect of making generated images in the same batch appear similar (e.g. they could all have a similar tint). Once we switched to the gradient penalty loss, we removed batch normalization in the discriminator model as suggested by the authors.

The generator model encodes the condition and noise by passing each through separate dense layers, then combines them by concatenating the encoded versions and passing that vector through a final dense layer. We then reshape the resulting vector and pass it through several deconvolutional (conv2d_transpose) layers to produce the generated image. The Discriminator model passes the input image through several convolutional layers and the encoded image is combined with the condition vector by concatenating encoded (through dense layers) version of these vectors together. The concatenated encodings are then passed through dense layers to get a real valued scalar output. We found that the use of leaky ReLU as the activation function helped the model perform better as fewer neurons would "die" while training. As discussed in the results section, replacing our standard convolutional blocks with Residual blocks improved image quality.

4.2 Loss functions

To start, we tried several loss functions on an **unconditioned** version of the problem where $\hat{y}$ indicates whether the example is from the real or generated distribution. The first loss we tried was the non-saturating version of the vanilla GAN loss (5):

$$J^{(G)} = -\mathbb{E}[\log(D(G(z)))] \tag{1}$$

$$J^{(D)} = -\mathbb{E}[y_{real} \log(D(x))] - \mathbb{E}[(1 - y_{gen})] \log(1 - D(G(z))) \tag{2}$$

We found this to be very sensitive to hyperparameter choice and random initialization which was not desirable. Very low learning rates let the losses converge better but took too long to train while faster learning rates (> 0.0001) led to mode collapse very quickly. To remedy this, we switched to the Wasserstein loss which proved to be much less susceptible to training instability and less sensitive to hyperparameter selection. Wassterstein GAN loss used in our work is based off the approximation by Arjovsky et al (6). Wasserstein GAN attempts to minimize an approximation of the intractible earth mover distance between the distribution of real and generated images. One of the requirements of the approximation described in (6) is that the weights of a discriminator function D must lie in a 'compact' space and that D is a K-Lipshitz function (meaning the first derivative of the function is bounded everywhere to be less than a constant). The authors of the WGAN paper enforce this constraint by clipping the weights of the network D to be within bounds $[-c, c]$. Another difference is that the discriminator output can no longer be interpreted as a probability and is a real value.

$$J^{(G)} = -\mathbb{E}[D(G(z))] \tag{3}$$

$$J^{(D)} = \mathbb{E}[D(x)] - \mathbb{E}[D(G(z))] \tag{4}$$

The authors of the original WGAN paper recognize that weight clipping is a poor way to enforce a Lipschitz constraint as the choice of hyperparameter c can cause vanishing gradients. We experimented with this and found that just as the authors claimed, the GAN became more resistant to mode collapse (where the GAN generates very similar images) and the generator was able to learn well even when the discriminator loss was very low. This version of our unconditional GAN model was much less sensitive to learning rate and other hyperparameter changes. However, we found that this method still had some shortcomings. Our generated images stopped improving at around 35 epochs and though the images looked reasonable at this point, they deteriorated and suffered mode collapse within 50-60 epochs. We believe this could be due to a combination of the dataset not being well

structured (have similar layout and structure of images) and the flaws in the weight clipping method. As such, we implemented WGAN-GP and found it to be a better algorithm. WGAN-GP improves upon the traditional WGAN by enforcing the 1-Lipshitz contraint on the discriminator function with a gradient penalty instead of weight clipping. The equations used for implementing this are as follows:

$$\epsilon \sim U[0,1]$$

$$\tilde{x} = G(z)$$

$$\hat{x} = \epsilon x + (1-\epsilon)\tilde{x}$$

$$J^{(G)} = -\mathbb{E}[D(G(z))] \tag{5}$$

$$J^{(D)} = \mathbb{E}[D(\tilde{x}) - D(x) + \lambda(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2] \tag{6}$$

The authors of the WGAN-GP note that their method allows for the discriminator to learn more complex functions and reduces the vanishing/exploding gradient problem. We removed batch normalization in the discriminator model as recommended by the authors of the improved WGAN paper and used hyperparameters suggested in their paper: Adam optimizer with learning rate 0.0001, $\beta_1 = 0$, $\beta_2 = 0.9$, $\lambda = 10$. We train the discriminator for 5 steps for each train step for the generator. Using this approach, our model performance was improved significantly. Tweaking hyperparameters further did not produce significant wins.

To achieve better conditioning, we also implemented an AC-wGAN-GP based on (7) and (8). The loss function for this model consisted of the wGAN-GP loss functions with additional categorical cross-entropy loss terms corresponding to the discriminator miscategorization loss for class labels Y (8).

$$A \cdot (\mathbb{E}[\log p(y = Y \mid x)] + \mathbb{E}[\log p(y = Y \mid \tilde{x})]) \tag{7}$$

was added to the discriminator loss and

$$B \cdot (\mathbb{E}[\log p(y = Y \mid \tilde{x})]) \tag{8}$$

was added to the generator loss. Hyperparameters A and B were set to $A = 1$ and $B = 0.1$ as suggested by the source code of (7).

5 Results and Discussion

Our criteria in evaluating images were (1) diversity and quality within the same class; and (2) the ability to differentiate and identify images between each class. We therefore evaluated our images using the following methods: (1) qualitative visual inspection; (2) Inception score (9); and (3) using the classification accuracy of a separately trained classifier.

5.1 Qualitative evaluation of images

We used qualitative visual evaluation to determine the quality of images and the efficacy of conditioning in our models. Example images generated by the ResNet cWGAN-GP, cWGAN-GP and the AC-wGAN-GP models are shown in Figure 3. The ResNet model produced sharper images that were more pleasing to the eye. The AC-wGAN-GP model appeared to be better for conditioning, which makes sense, given that the AC-wGAN-GP loss functions contain additional categorical loss terms. Our images replicated some of the geometric structure of food (the pancakes are round and have sharp edges, and some of the tiramisu and apple pie appear as well-defined slices), but struggled with details (leaves in the Caesar salad are not well-formed, waffles lack consistent grid patterns).

5.2 Inception score

Inception score uses the output of an Inception network pretrained on ImageNet to evaluate the quality and diversity of the images.

$$\text{IS} = \exp\left\{\mathbb{E}_{\tilde{x}\sim p_g}[D_{KL}(p(y \mid \tilde{x}) \parallel p(y))]\right\} \tag{9}$$

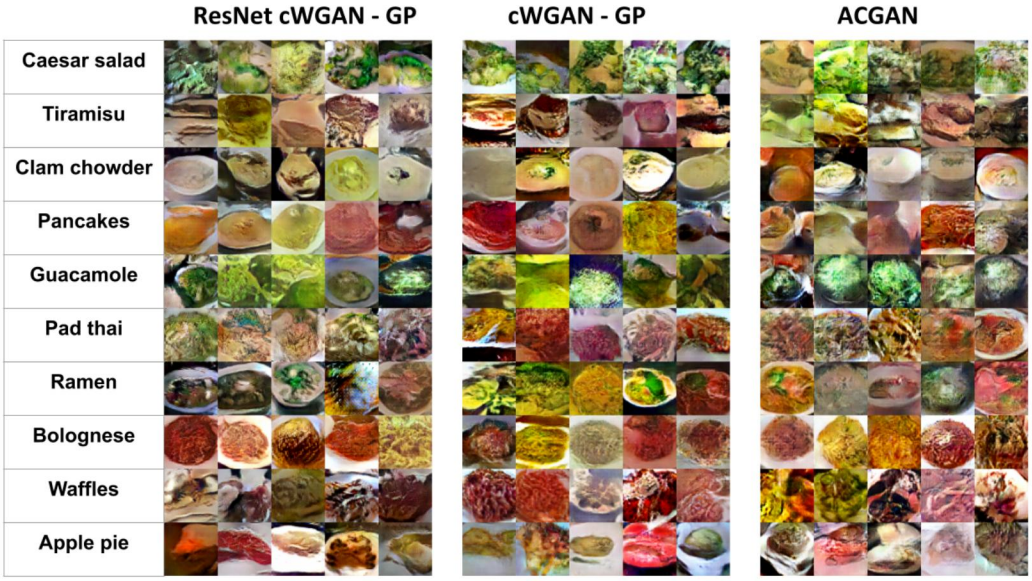

Figure 3: Example conditional images generated, in left-to-right order, by the ResNet cWGAN-GP, vanilla cWGAN-GP, and the AC-WGAN-GP, respectively.

$\tilde{x}$ is the image generated by the GAN, $p(y \mid \tilde{x})$ is the conditional probability distribution of the class of $\tilde{x}$ determined by the Inception network, and $p(y) = \int_{\tilde{x}} p(y \mid \tilde{x})p(\tilde{x})$ is the marginal distribution. Higher Inception scores correlate with better image quality and diversity, because good images should be classified as a single class with high probability, making the entropy of $p(y\tilde{x})$ be low, while the entropy of $p(y)$ should be high for a diverse set of $\tilde{x}$, making $D_{KL}(p(y \mid \tilde{x}) \parallel p(y))$ large. Conversely, GANs that produce low-quality poorly defined images or suffer from mode collapse should tend to have low Inception scores.

Inception score is an imperfect metric for evaluating conditional GANs. Diversity does not translate well to conditional GANs, since conditioning reduces the entropy of $p(y)$. Furthermore, there is no clear-cut way of combining Inception scores across the different conditions. We chose to calculate Inception scores separately for each class and report the average across classes.

Despite these limitations, Inception score is useful for measuring the quality/sharpness of images and the amount of mode collapse. Inception scores plotted during training are shown in Figure 4 Inception scores for the ResNet typically were higher than those for the AC-WGAN and vanilla GAN. This appeared to be due to the sharper, better-defined images generated by the ResNet model. The Inception scores declined near the end of training for the ResNet, corresponding to visibly grainier images. Inception scores for all of our models varied between 4 and 4.5. For reference, the maximum achievable Inception score is 1000, and state-of-the-art Inception scores are around 10 (10). As a comparison, the Bar El *et al.* paper achieved an Inception score of around 4.55 (2) for their food images.

5.3 Classifier accuracy to evaluate conditioning

We trained a separate CNN classifier using the same subset of the Food-101 dataset, and used the classification accuracy of that classifier as a metric for the effectiveness of conditioning. This is a simple metric – if the classifier accurately classifies the generated images, then the GAN-generated images are likely well-formed and from the correct class.

Despite our best efforts, our classifier overfit to the training data (90+% training accuracy but 65% test accuracy). Thus, we expected that the classifier will perform relatively poorly when applied to the generated images.

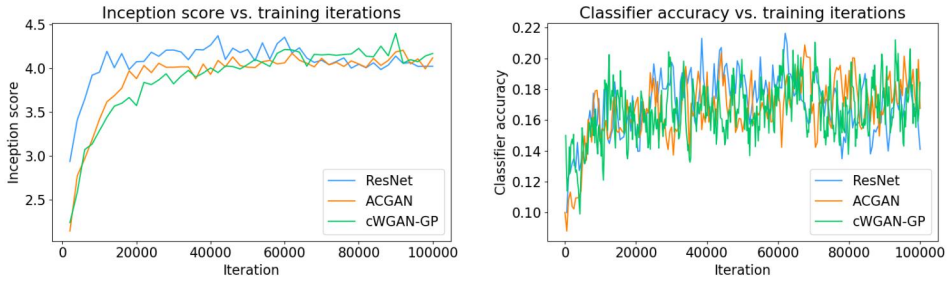

Figure 4: Inception score and classifier accuracy for each of the models.

Classifier accuracy for the three models are shown in the right panel of Figure 4. Indeed, classifier performance is poor. Classifier accuracy for all three models started at 10% (no better than random chance) at the beginning of training, and reached around 20% by the end of training. Given the limitations of our classifier, we conclude that the conditioning in our models perform better than random chance.

6 Conclusion/Future Work

cWGAN-GP with Residual blocks proved to have the best performance in terms of our metrics as well as producing sharper, more well defined images. We believe this was because of the improved stability combined with the skip connections and more complex networks.

Incorporating the condition had the most room for improvement so we would like to explore better ways of using the category labels. Possible approaches could include spectral normalization (11), projection discriminator (12) and stack GANs with conditioning augmentation (13).

7 Contributions

Eric: Data input pipelines and augmentation. Preprocessed dataset into the right format, etc. Set up Inception score metrics.
Sudip: Model and loss function selection and implementation. Setting up tensorflow model graph. Set up classifier for evaluation.

References

[1] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," *arXiv e-prints*, Nov. 2014.

[2] O. Bar El, O. Licht, and N. Yosephian, "GILT: Generating Images from Long Text," *arXiv e-prints*, p. arXiv:1901.02404, Jan 2019.

[3] Y. Ito, W. Shimoda, and K. 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*, ser. CEA/MADiMa '18. New York, NY, USA: ACM, 2018, pp. 71–74. [Online]. Available: http://doi.acm.org.stanford.idm.oclc.org/10.1145/3230519.3230598

[4] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101 – mining discriminative components with random forests," in *European Conference on Computer Vision*, 2014.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[6] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *arXiv e-prints*, p. arXiv:1701.07875, Jan 2017.

[7] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved Training of Wasserstein GANs," *arXiv e-prints*, p. arXiv:1704.00028, Mar 2017.

[8] A. Odena, C. Olah, and J. Shlens, "Conditional Image Synthesis With Auxiliary Classifier GANs," *arXiv e-prints*, p. arXiv:1610.09585, Oct 2016.

[9] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved Techniques for Training GANs," *arXiv e-prints*, p. arXiv:1606.03498, Jun 2016.

[10] S. Barratt and R. Sharma, "A Note on the Inception Score," *arXiv e-prints*, p. arXiv:1801.01973, Jan 2018.

[11] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.

[12] T. Miyato and M. Koyama, "cgans with projection discriminator," *arXiv preprint arXiv:1802.05637*, 2018.

[13] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5907–5915.