Generating Object Stamps

Youssef A. Mejjati
University of Bath

Zejiang Shen
Brown University

Michael Snower
Brown University

Aaron Gokaslan
Brown University

Oliver Wang
Adobe Research

James Tompkin
Brown University

Kwang In Kim
UNIST

Figure 1: Given a user-provided background image, object class (giraffe), and bounding box (top left), we composite objects with diverse shapes and textures (*right*). *Bottom:* We combine multiple object classes across scenes and match background illumination.

Abstract

We present an algorithm to generate diverse foreground objects and composite them into background images using a GAN architecture. Given an object class, a user-provided bounding box, and a background image, we first use a mask generator to create an object shape, and then use a texture generator to fill the mask such that the texture integrates with the background. By separating object insertion into these two stages, we show that our model improves the realism of diverse object generation that also agrees with the provided background image. Our results on the challenging COCO dataset show improved overall quality and diversity compared to baseline object insertion approaches.

1. Introduction

Compositing objects into images is a common editing task. Given a database of images of a target object class and a specific background image, the task typically proceeds in three steps: 1) Find an instance of the object in a suitable pose and under similar lighting to the background image; 2) define the location and size of the object in the background image; and 3) composite the object onto the background in a visually-consistent way. Often this process can be cumbersome, and requires time and skill.

Paper accepted to the CVPR 2020 AI for Content Creation workshop.

Our goal is to reduce the burdens of steps 1 and 3 to create a simple user interface for object compositing. We begin with a database of images with object masks. We wish to learn how to represent the object class' appearance variation, and how to match the appearance with background scenes. This would let the user generate an object by simply dragging a bounding box over novel background images, and then by sampling multiple shapes and textures from the learned space of the object class.

We decompose the generation process into two steps. First, we synthesize foreground masks (representing the shape of the object) conditioned on a user specified bounding box. Next, we generate realistic texture within the mask via *conditioning on the shape mask and background image*. This allows us to fix the shape while varying the texture (or vice versa). At each stage, we inject randomness expressed by a latent random variable, which allows the generation of multiple shape masks and textures given the same bounding box coordinates.

2. Related Work

Singh *et al.*'s FineGAN [14] can generate convincing images of a class by separating the operation into sequential background and foreground generation. Composite-GAN [9] and LRGAN [19] similarly recursively generate the background and foreground, but do not disentangle shape and appearance in the same way as FineGAN. Contemporaneously, Turkoglu et al. [17] generate instances from bounding boxes, but their mask prediction is deterministic and so is less useful. Furthermore, their generation pipeline is class conditioned and cannot be straightforwardly trained if only data from one class is available. Our method is inspired by these methods, as they demonstrate the power of decomposed generation. However, our work differs in two key ways: First, we generate a foreground with respect to an *existing* background. Second, our method allows for control over the location and scale of the generated foreground object conditioned on the background content.

Lee et al. [10] learns to generate diverse shapes, and learns their location distribution. However, this method focuses only on shape and does not learn to texture. Furthermore, it does not allow users to place objects interactively on a given background. Finally, training the model requires semantic segmentation maps at training time; in contrast, our algorithm only requires binary masks.

Most similar to our work, Hong *et al.* [4] 'complete' a user

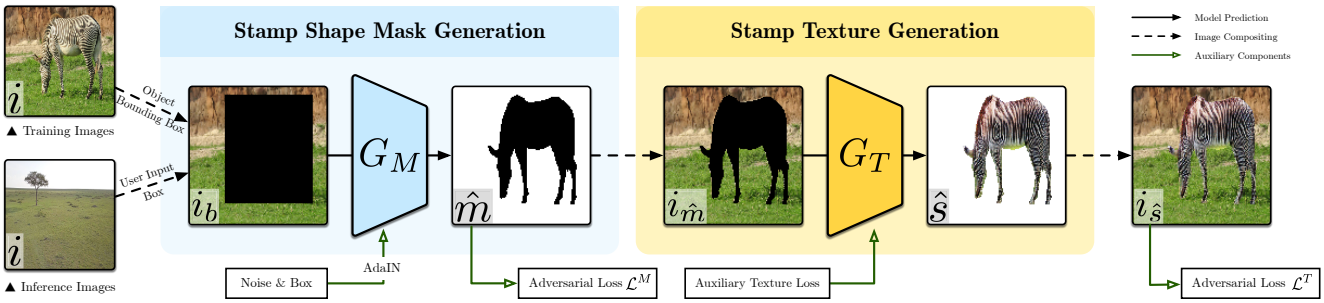

Figure 2: *Overview of our image generation pipeline.* Given an input image i, a bounding box is cut out to create i_b, and this is then fed into the mask generator G_M. G_M creates a plausible object mask from the desired category (in this case, zebra). Our texture generator then fills in the mask to produce a final composited image, while a texture discriminator ensure image realism.

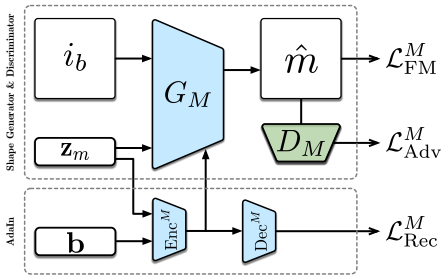

Figure 3: Our stamp shape generation model.

bounding box by generating an object instance. Our model has three key differences: we can generate multiple object shapes and textures, we use the entire background for conditioning to improve harmonization, we do not require a semantic segmentation map to generate good shapes. We compare our result directly with Hong *et al.*'s approach, and show that we can generate results with higher quality and instance diversity.

3. Method

Our model takes as input a background image i drawn from a domain $\mathcal{I}$, a bounding box $\mathbf{b}\in\mathbb{R}^4$ containing rectangle vertices. From $\mathbf{b}$, we construct a binary bounding box image b of same size as i with the region inside the box set to 1, and 0 otherwise. The goal is to generate an object stamp inside the bounding box and to composite it with the background image i (Figure 2). We achieve this by first generating a stamp mask $\hat{m}$, and then generating a textured stamp $\hat{s}$ such that when composited into the final image, $i_{\hat{s}}$ is indistinguishable from images in $\mathcal{I}$, where $i_{\hat{s}} = i\odot(1-\hat{m})+\hat{s}\odot\hat{m}$, and where $\odot$ is the element-wise product.

3.1. Stamp shape generation model

We train a generator G_M that takes the background image as input and is conditioned on a bounding box, and a random vector $\mathbf{z}_m\sim\mathcal{N}(0,I)$. Importantly, we train the generator on images that contain object instances and their respective masks. Therefore, to allow us to use background content at test time that does *not* contain the object in question, we first zero-out the bounding box region $i_b = i\odot(1-b)$. This prevents the network from trivially learning to segment existing instances in the training data.

The generator G_M produces a binary mask $\hat{m}$ for the shape of the stamp object inside the bounding box region, *e.g.*, $\hat{m} = G_M(i_b,\mathbf{z}_m)$. We train G_M adversarially: G_M attempts to generate realistic shapes to fool a discriminator D_M, while D_M attempts to classify generated masks separately from real training data masks. D_M is a CNN which takes as input the shape mask and corresponding bounding box: $D_M(m,b)$. We use a hinge-GAN loss $\mathcal{L}^M_{\text{Adv}}$ to train both G_M and D_M for better stability [11, 15, 12]:

$$\mathcal{L}^M_{\text{Adv}}(G_M,D_M) = \mathbb{E}_m[\min(0,D_M(m,b)-1)] + \mathbb{E}_{\hat{m}}[\min(0,-D_M(\hat{m},b))-1], \quad (1)$$

where m is a ground-truth shape mask and $\hat{m}$ is generated.

Next, we describe how we use adaptive instance normalization (AdaIN) [5] in G_M to condition the network on the input noise $\mathbf{z}_m$. In our case, we wish to inject the bounding box $\mathbf{b}$ and the latent vector $\mathbf{z}_m$ during shape generation. For this, similar to prior work [5], we use a small fully-connected feed-forward network (MLP) encoder Enc^M to take input $\mathbf{b}$ and $\mathbf{z}_m$ and predict affine transformation parameters $\text{Enc}^M(\mathbf{b},\mathbf{z}_m)$ for the instance normalization layers.

One issue we found is that AdaIN can learn to ignore $\mathbf{z}_m$ by using only $\mathbf{b}$, which reduces diversity in generation. To overcome this, we propose a reconstruction loss $\mathcal{L}^M_{\text{Rec}}$ over $\mathbf{z}_m$ via an MLP decoder complement Dec^M to Enc^M:

$$\mathcal{L}^M_{\text{Rec}}(\text{Enc}^M,\text{Dec}^M) = \frac{1}{|\mathbf{z}_m|}\|\mathbf{z}_m - \text{Dec}^M(\text{Enc}^M(\mathbf{b},\mathbf{z}_m))\|_1^2. \quad (2)$$

Unlike prior work [21], Dec^M decodes the latent vector from $\text{Enc}^M(\mathbf{b},\mathbf{z}_m)$ rather than the output mask $\hat{m}$, which we found to perform better in our experiments. This loss directly enforces that the AdaIN parameters depend on the random vector $\mathbf{z}_m$, and so helps to maintain diversity. This yields diverse results, but we found that the masks lacked fine detail. Therefore, we add a deep feature matching loss [13, 20] which enhances sharpness by enforcing that real and generated images elicit similar feature responses in each layer l of the discriminator $D^{(l)}_M$:

$$\mathcal{L}^M_{\text{FM}}(D_M) = \mathbb{E}_{\hat{m},m}\sum_{l=1}^{L}||D^{(l)}_M(\hat{m},b) - \bar{D}^{(l)}_M||_2^2, \quad (3)$$

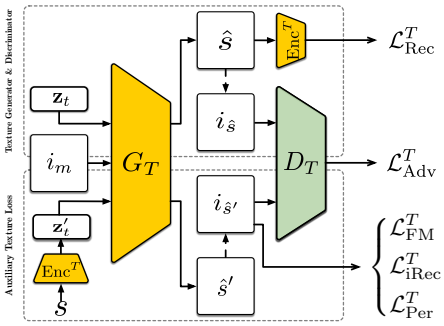

Figure 4: Our stamp texture generation model.

where $\bar{D}^{(l)}_M$ is the moving average of features in layer l.

Our combined learning objective for shape generation is a weighted combination of the aforementioned losses:

$$\mathcal{L}^M = \mathcal{L}^M_{\text{Adv}} + \lambda^M_{\text{FM}}\mathcal{L}^M_{\text{FM}} + \lambda^M_{\text{Rec}}\mathcal{L}^M_{\text{Rec}}, \tag{4}$$

where $\lambda^M_{\text{FM}}, \lambda^M_{\text{Rec}} \geq 0$ are hyperparameters set to 10.

3.2. Stamp texture generation model

The goal of the texture generator G_T is to create realistic textures that match both the pose in the shape mask and the lighting in the background. Given a generated shape mask $\hat{m}$, we first zero-out the shape from the input image to mark the area that needs texturing: $i_{\hat{m}} = i \odot (1 - \hat{m})$. Then, the stamp texture generation G_T synthesizes the texture inside the empty region while still having access to the texture of the surrounding background: $\hat{s} = G_T(i_{\hat{m}}, \mathbf{z}_t)$. $\hat{s}$ is the generated object stamp image, and $\mathbf{z}_t \sim \mathcal{N}(0, I)$ is a random vector that adds stochasticity for diverse generation. For G_T, we use a BicycleGAN-like [22] architecture to preserve both texture quality and diversity. Similar to G_M, we train G_T adversarially via a texture discriminator D_T.

The input to the discriminator D_T is the channel-wise concatenation of the real image and mask tuple (i, m) or the fake equivalent $(i_{\hat{s}}, \hat{m})$. Passing the mask to D_T tells the discriminator to expect an instance of the object at that location. This is important as, at training time, the background image $i_{\hat{s}}$ may contain *multiple* instances of the target class, and so D_T would be satisfied by the generator filling in the mask with background instead of object.

Unlike D_M, we do not use a moving average feature matching loss as we have access to the ground-truth texture. Instead, we use a CNN to encode the ground-truth texture of the foreground $s = i \odot m$ as a latent vector $\mathbf{z}'_t = \text{Enc}^T(s)$, and then use $\mathbf{z}'_t$ and m to generate another stamp texture $\hat{s}' = G_T(i_m, \mathbf{z}'_t)$ and corresponding stamped image $i_{\hat{s}'} = i \odot (1 - m) + \hat{s}' \odot m$. As $\mathbf{z}'_t$ depends on s, this indirectly conditions $i_{\hat{s}'}$ on s, and allows us to apply a feature matching loss without losing detail:

$$\mathcal{L}^T_{\text{FM}}(D_T) = \mathbb{E}_{i_{\hat{s}'}, i} \sum_l ||D^{(l)}_T(i_{\hat{s}'}, m) - D^{(l)}_T(i, m)||^2_1. \tag{5}$$

Here, $D^{(l)}_T$ is the output of the l-th layer in D_T. We also apply an additional L_1 image reconstruction loss $\mathcal{L}^T_{\text{iRec}}$ to aid in the description of the texture corresponding to $\mathbf{z}'_t$: $\mathcal{L}^T_{\text{iRec}} = \frac{1}{|i|}\sum_{l=1} ||i_{\hat{s}'} - i||^2_1$.

As an extra constraint on $\mathbf{z}'_t$, we wish for its distribution $\mathcal{Z}'_t$ to be similar to the distribution of $\mathbf{z}_t$, such that samples at test time remain realistic. Thus, we use the re-parametrization trick [8] on Enc^T and add a KL-divergence loss $\text{KL}(\mathcal{Z}'_t || \mathcal{N}(0, I))$ to promote distribution consistency.

Texture architecture: Compared to our mask architecture, we make five additional changes to our texture model. First, unlike for masks, we adopt a BicycleGAN-like architecture and hence we do not use AdaIN to inject auxiliary information into the generator. Second, to make sure that the latent vector $\mathbf{z}_t$ is not ignored, we penalize a latent texture reconstruction loss on $\mathbf{z}_t$ via an encoder on $\hat{s}$:

$$\mathcal{L}^T_{\text{Rec}}(\text{Enc}^T) = \frac{1}{|\mathbf{z}_t|} ||\mathbf{z}_t - \text{Enc}^T(\hat{s}))||^2_1. \tag{6}$$

Similar to Zhu et al. [22], we do not update Enc^T when propagating the gradients from Eq. 6. This avoids that Enc^T hides information in the data, making it easy to reconstruct [2].

Third, to aid realism in both $i_{\hat{s}}$ and $i_{\hat{s}'}$, we use both to train G_T and D_T. As such, the adversarial loss $\mathcal{L}^T_{\text{Adv}}$ becomes:

$$\begin{aligned}\mathcal{L}^T_{\text{Adv}}(G_T, D_T) = \; & \mathbb{E}_i[\min(0, D_T(i, m) - 1)] + \\ & \mathbb{E}_{i_{\hat{s}}}[\min(0, -D_T(i_{\hat{s}}, m) - 1)] + \\ & \mathbb{E}_{i_{\hat{s}'}}[\min(0, -D_T(i_{\hat{s}'}, m) - 1)].\end{aligned} \tag{7}$$

Fourth, we add a perceptual loss [6] to help recreate fine textural details, *e.g.*, the tail of a giraffe. This loss uses a pre-trained VGG16 network to extract features for two image instances, then enforces that their feature activations are similar:

$$\mathcal{L}^T_{\text{Per}}(G_T) = \frac{1}{N} \|\phi(i) - \phi(i_{\hat{s}'})\|^2_1, \tag{8}$$

where $\phi(i)$ is the third layer output the VGG16 network, and extracts N features from i.

Fifth, and finally, we add Gaussian noise $\mathcal{N}(0, I)$ to the texture decoder *a la* StyleGAN [7]. This helps the diversity of the results and improves image generation quality.

The overall texture generation training loss $\mathcal{L}^T$ is:

$$\begin{aligned}\mathcal{L}^T = \; & \mathcal{L}^T_{\text{Adv}} + \lambda^T_{\text{Rec}}\mathcal{L}^T_{\text{Rec}} + \lambda^T_{\text{KL}}\text{KL}(\mathcal{Z}'_t || \mathcal{N}(0, I)) \\ & + \lambda^T_{\text{FM}}\mathcal{L}^T_{\text{FM}} + \lambda^T_{\text{Per}}\mathcal{L}^T_{\text{Per}} + \lambda^T_{\text{iRec}}\mathcal{L}^T_{\text{iRec}},\end{aligned} \tag{9}$$

where λ^T_{Rec}, λ^T_{KL}, λ^T_{FM},λ^T_{Per},and $\lambda^T_{\text{iRec}} \geq 0$ are hyperparameters. λ^T_{Rec} and the KL divergence increase diversity, while the λ^T_{FM}, λ^T_{Per}, and λ^T_{iRec} improve the quality of the generated texture. All λ hyper-parameters are set to 10 apart from λ^T_{KL} which is set to 0.05.

4. Experiments

Datasets: We extract two animal classes from COCO: *Giraffe* (2,205 images), *Zebra* (2,306 images). We exclude all instances that are smaller than 1% of the entire image, all that contain multiple separate components, and all that intersect any image border. We collect background images for stamping from the Internet by searching relevant queries under a 'free to use and modify' license.

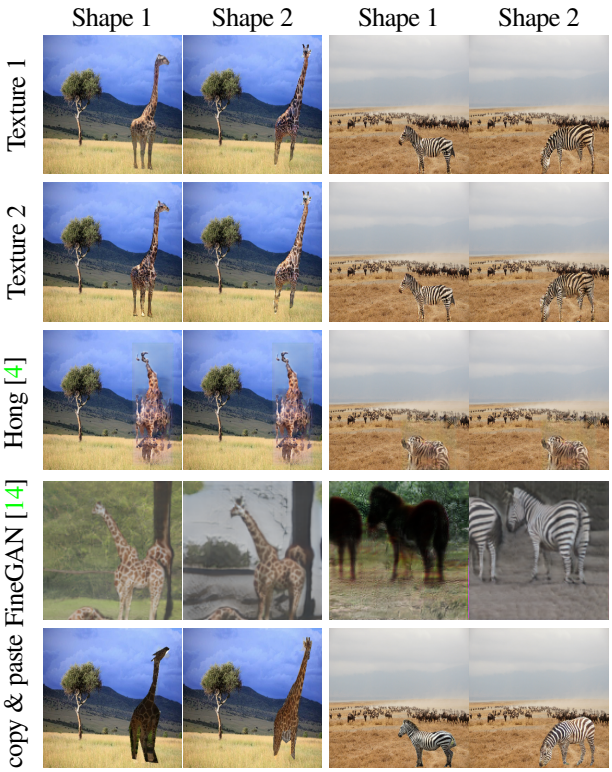

Figure 5: COCO classes *Giraffe* and *Zebra*, showing our diverse generated shapes (in columns) and diverse textured results (row 1 and 2), deterministic images from Hong et al.'s algorithm [4] (row 3), FineGAN's whole-image generation [14](row 4), and Copy and Paste + deep harmonization baselines (row 5).

4.1. Method comparisons

We are not aware of any methods that addresses our exact problem, and so we adapt related methods for comparison.

Semantic image manipulation of Hong et al. [4]: This approach is designed to use semantic segmentation masks, whereas our application uses binary masks instead. In this setting, their method produces less distinct shapes and blurrier textures than our approach (Fig. 5). Further, Hong et al.'s algorithm is deterministic: it insert only one instance given a bounding box, whereas our approach can create multiple shapes and textures from the same bounding box. Finally, Hong et al. generate the entire bounding box, which requires also generating background regions to match seamlessly, which is challenging.

FineGAN [14]: This whole-image approach decomposes generation into first generating the background and then the foreground. It produces convincing results on datasets contianing a hierarchical structure such as bird images of CUB [18]. However, when trained on our classes from COCO, performance tend to decrease (Figs. 5), or collapse altogether. Further, while FineGAN allows for stochastic texture generation, it cannot generate localized instances for user control over the scene layout, *e.g.*, generated foregrounds are often in the center of the image.

Table 1: Kernel Inception Distance$\times 100 \pm$std$\times 100$ across all datasets used. Lower is better.

Models	*Giraffe*	*Zebra*
SIM	8.43 ± 0.59	6.78 ± 0.79
Ours	$\mathbf{4.87 \pm 0.45}$	$\mathbf{5.12 \pm 0.67}$
FineGan	17.66 ± 0.86	12.51 ± 0.60
Ours	$\mathbf{1.37 \pm 0.24}$	$\mathbf{1.28 \pm 0.24}$

Copy and Paste + Deep Harmonization [16]: We also compare with a copy and paste baseline. First, we find masks in the training set that most resemble generated masks from our algorithm. Second, to make this baseline more realistic, we use a state of the art harmonization approach [16]. Results in Figures 5 show that this baseline often struggles to form a convincing composite. Furthermore, this approach has three other limitations: 1) Without using our model's generated mask to query the database, the user would be required to provide an actual mask rather than a bounding box; 2) It requires user time to search through the training dataset; and 3) It is not able to disentangle shape from texture.

4.2. Quantitative Evaluation

We evaluate using Kernel Inception Distance KID [1], which gives more consistent results than FID, especially with small numbers of generated samples [3].

Semantic image manipulation of Hong et al. [4]: As this approach is deterministic, we generate one instance per bounding box and per scene. We do the same for our method using the same bounding boxes and background images. We report KID results on 50 random subsamples in Table 1. Our approach consistently performs better, which agrees with the visual results (Fig. 5).

FineGAN [14]: We trained FineGAN on the COCO classes using our image resolution (256×256), which is higher than used in their paper. At this increased resolution, FineGAN mode collapsed, and so we used their original resolution (128×128) and bilinearly upsample the generated images to 256×256. FineGAN generates an entire image, so a comparison using our composited foregrounds would not be fair to it. Instead, we compute KID using only generated foregrounds. Table 1 shows the corresponding scores. FineGAN produces substantially worse KID scores than our method (Figure 5). Running the algorithm on classes such as 'giraffe' and 'zebra' results in a significant reduction in generated diversity in that the same instance is generated most of the time. Further, resizing the images to 256x256 due to mode collapses in training at higher resolutions causes a lack of fine detail that is captured by KID. Finally, some foreground generations are not successful (third column).

5. Conclusion

We generate object stamps by splitting shape and texture generation, which achieves detail and diversity in both aspects. Further, we condition generation on background content with a way to train on pre-existing segmentation datasets which *include* object instances already but do not confuse the generator. This provides flexible generation via a simple bounding box interface.

References

[1] M. Bińkowski, D. Sutherland, M. Arbel, and A. Gretton. Demystifying MMD GANs. In *ICLR*, 2018. 4

[2] Casey Chu, Andrey Zhmoginov, and Mark Sandler. Cyclegan, a master of steganography. In *NeurIPS, workshop on Machine Deception*, 2017. 3

[3] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Klambauer. GANs trained by a two time-scale update rule converge to a Nash equilibrium. In *NeuIPS*, 2017. 4

[4] S. Hong, X. Yan, T. Huang, and H. Lee. Learning hierarchical semantic image manipulation through structured representations. In *NeurIPS*, 2018. 1, 4

[5] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 2

[6] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 3

[7] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. *CVPR*, 2019. 3

[8] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014. 3

[9] H. Kwak and B.-T. Zhang. Generating images part by part with composite generative adversarial networks. *arXiv preprint arXiv:1607.05387*, 2016. 1

[10] Donghoon Lee, Sifei Liu, Jinwei Gu, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Context-aware synthesis and placement of object instances. In *Advances in Neural Information Processing Systems*, pages 10393–10403, 2018. 1

[11] J. H. Lim and J. C. Ye. Geometric GAN. *arXiv preprint arXiv:1705.02894*, 2017. 2

[12] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. 2018. 2

[13] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *NeuIPS*, 2016. 2

[14] K. K. Singh, U. Ojha, and Y. J. Lee. Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. In *CVPR*, 2019. 1, 4

[15] D. Tran, R. Ranganath, and D. M. Blei. Deep and hierarchical implicit models. *arXiv preprint arXiv:1702.08896*, 7, 2017. 2

[16] Y.-H. Tsai, X. Shen, Z. Lin, K. Sunkavalli, X. Lu, and M.-H. Yang. Deep image harmonization. In *CVPR*, 2017. 4

[17] Mehmet Ozgur Turkoglu, William Thong, Luuk Spreeuwers, and Berkay Kicanaoglu. A layer-based sequential framework for scene generation with gans. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:8901–8908, Jul 2019. 1

[18] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 4

[19] J. Yang, A. Kannan, D. Batra, and D. Parikh. LR-GAN: Layered recursive generative adversarial networks for image generation. In *ICLR*, 2017. 1

[20] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018. 2

[21] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman. Toward multimodal image-to-image translation. In *NeuIPS*, 2017. 2

[22] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *NeurIPS*, 2017. 3