This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Collaging Class-specific GANs for Semantic Image Synthesis

Yuheng Li[1] Yijun Li[2] Jingwan Lu[2] Eli Shechtman[2] Yong Jae Lee[1] Krishna Kumar Singh[2]
[1]University of Wisconsin-Madison [2]Adobe Research

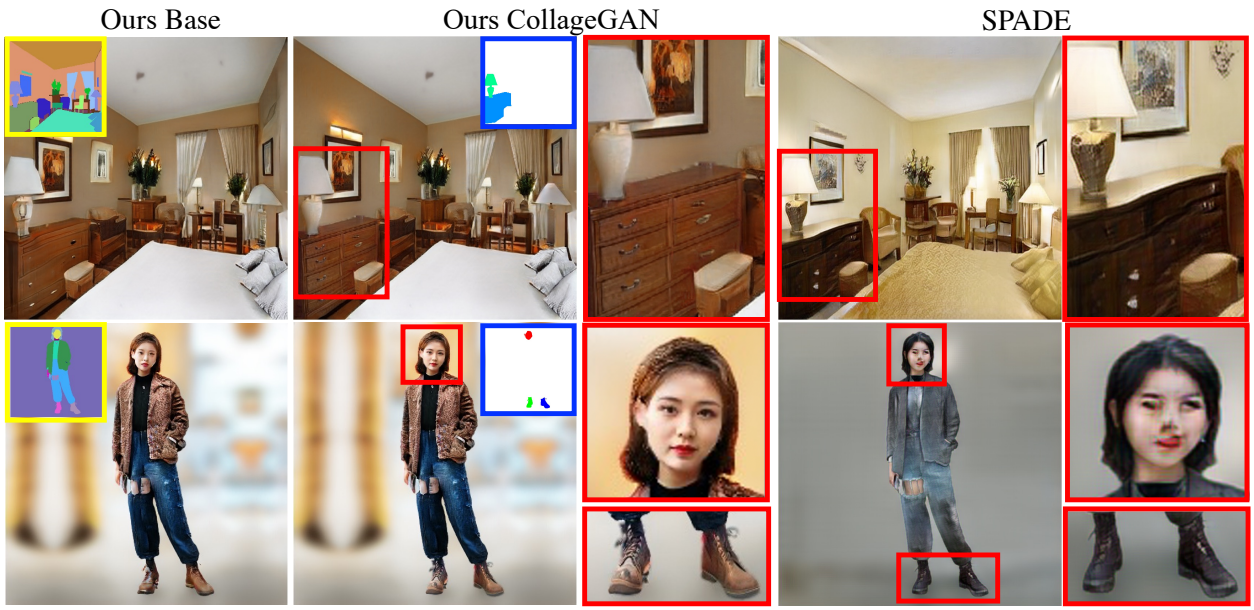

Figure 1. High resolution image generation conditioned on a segmentation map (yellow inset). Our base model (1st column) generates more realistic results than SPADE [24] (4-5th column). We further improve the quality of results by using our class-specific generators to generate foreground objects or parts and compose them on the image generated by our base model. The segmentation maps in 2nd column (blue insets) show the parts that are modified by our bank of GANs and the zoomed-in results are shown next to each image.

Abstract

We propose a new approach for high resolution semantic image synthesis. It consists of one base image generator and multiple class-specific generators. The base generator generates high quality images based on a segmentation map. To further improve the quality of different objects, we create a bank of Generative Adversarial Networks (GANs) by separately training class-specific models. This has several benefits including – dedicated weights for each class; centrally aligned data for each model; additional training data from other sources, potential of higher resolution and quality; and easy manipulation of a specific object in the scene. Experiments show that our approach can generate high quality images in high resolution while having flexibility of object-level control by using class-specific generators.

1. Introduction

Image generation has been explored extensively in both the unconditional [8, 18, 19, 32] and conditional [15, 37, 33, 17, 3, 24] settings. In the unconditional setting, an image is generated by randomly sampling a latent code. With the advent of StyleGAN2 [32], for some classes such as faces, the generated images are almost indistinguishable from real images. In the conditional setting, an image is generated based on some input conditioning signal such as another image, a segmentation map, or other priors. Most notably, SPADE [24] (i.e., spatially-adaptive normalization) significantly boosted image quality for semantic image generation. This normalization and its variants have been used as standard building blocks in many following works [35, 25, 31, 30]. However, the image quality of conditional GANs is still inferior to that of unconditional GANs, especially StyleGAN2. Also, most approaches output 256×256 resolution images, which is not high enough for many real world applications, and their generation quality is inferior for higher resolutions.

In this work, we target high resolution image generation conditioned on input segmentation maps (yellow insets in

Fig. 1, first column). Contrary to most prior work which typically follow the SPADE architecture design, we explore a new direction. First, we develop a conditional version of StyleGAN2. Our generator backbone builds upon the original StyleGAN2 architecture, which can generate better quality images and is more easily scaled to high resolutions. Specifically, we use an encoder to extract hierarchical feature representations from the input segmentation map and use it to modulate the StyleGAN2 generator. The resulting generator is less memory intensive and faster to train compared to SPADE based approaches.

Second, instead of using a single generator, we use *multiple class-specific generators* to improve the quality of small foreground objects. There are several advantages of using different models for different classes: (i) Class-specific generators with dedicated weights learned for each foreground class have the capability to generate more details. Imagine a bedroom filled with objects of different scales such as bed, lamp, table, chair, chest and others (Fig. 1, first row). A single GAN might allocate most of its capacity toward generating larger content such as floor, walls and bed since they contribute most to the overall realism, and therefore neglect the details of smaller objects such as lamps and chests. Using separate generators for smaller objects result in more textural details and better shape integrity in those regions (Fig. 1, columns 2-3). (ii) Class-specific generators can be trained with image crops that always align the objects to the center. As shown in [19, 32], spatial alignment is often crucial for high quality generation. On the other hand, a single GAN generating an entire scene at once needs to deal with objects appearing in arbitrary locations which is more difficult to learn. (iii) Class-specific generators can benefit from more training data, which is especially important for more rare classes. Other than the scene training set from which all foreground objects are extracted, one can leverage separate class-specific datasets for training each foreground generator. (iv) Class-specific generators enable more applications. For example, we can generate out-of-distribution scene images such as a car on a sidewalk (Fig. 8), and we can also easily change the appearance of one object instance in the scene without modifying the rest of it (Fig. 7).

However, naïvely training separate generators and combining their results does not generate satisfactory results. It can lead to appearance inconsistency among generated objects in the same scene; e.g., a generated foreground object might not have compatible colors and perspective with its background or in the context of other objects. Training all models together based on shared global information will address the inconsistency issue, but it is extremely computationally demanding and cannot scale. Instead, we provide the result of our single GAN base model as input to each class-specific generator since it captures the global knowledge of the scene and constrains the foreground object appearance to be harmonious with each other. Although our focus is on generating complex scenes, our approach can also be used to generate a single object class with multiple complex parts like humans (Fig. 1, second row). We demonstrate this by training separate class-specific generators for faces and shoes.

Contributions. (1) A general framework for high quality conditional generation of complex scenes by training multiple class-specific generators; (2) A powerful StyleGAN2-based conditional base generator; (3) We demonstrate state-of-the-art image quality synthesized by our model against existing methods and show some potential applications.

2. Related work

High Resolution Semantic Image Generation. GANs [9] have become one of the most promising models to generate photorealistic images. Many different architectures [4, 39, 37, 33, 18, 3, 19, 32] and training techniques [1, 11, 23] have been proposed to explore different tasks [4, 39, 37, 33], improve generation quality [3, 19, 32], and stabilize training [1, 11, 23]. There are also many great works exploring conditional image generation [37, 33, 17, 3, 24, 31, 30]. In particular, segmentation map to image generation [33, 24, 31, 30] is popular due to its wide applications like flexible content creation for image editing. Although these works generate impressive results at lower resolutions, their higher resolution results for complex settings such as indoor scenes or full human bodies are often inferior and less detailed. Even at lower resolutions, the quality of foreground objects can be low due to the model focusing more on the background, which typically has more pixels than the foreground. The current state-of-the-art method for high resolution image generation is StyleGAN2 [19, 32]. However, it is an unconditional model which lacks controllability over the image generation process. Some recent concurrent works [20, 27] modify the StyleGAN2 architecture for conditional image generation for the specific domain of human bodies. In contrast, we propose a general purpose approach to condition StyleGAN2 on segmentation map inputs of various scene categories (e.g., bedrooms and street scenes) and complex objects (e.g., humans).

Image generation through unshared weights. Most image generation models have a monolithic architecture [33, 18, 3, 24, 19, 32], which outputs a final image without specialized weights for different classes or semantic regions. Others [36, 14, 21, 10, 13, 29, 31] model an image via different components, e.g., by having global and local branches [14, 21, 10, 13] with the local branch being specialized for different objects/parts. However, these models are usually face specific [14, 21, 10]. The work of [13, 36, 29] propose to separate background and foreground generation. But they can only generate single object

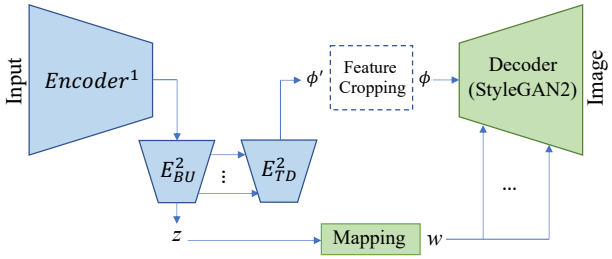

Figure 2. **The common architecture used for both the base model and the class-specific models** We modify the StyleGAN2 architecture by replacing its input constant with the output from the encoder. z is also conditioned on the encoder output.

images and do not have dedicated generators for specific foreground categories. [31] shares a similar idea with us. However, instead of having separate modules for different classes, it only has a few separate convolution layers for each class at the end of the generator, which means that it is deprived of our advantages of aligning training data, having an entire network dedicated to a class, and using data from other sources. Moreover, the model is very memory extensive and time consuming to train, as all the classes have to be trained together in an end-to-end fashion. In contrast, our class-specific generators can be trained for selected classes independently on separate GPUs in parallel.

3. Approach

We aim to generate high-quality scene images based on input segmentation maps by using a powerful base generator that generates an entire scene without differentiating objects and parts, and multiple class-specific generators that improve the appearances of small foreground objects. Both types of generators use the same architecture design with minor changes. We first introduce our new architecture (Section 3.1). Then, we describe our training (Section 3.2) and inference pipelines (Section 3.3).

3.1. Architecture

Fig. 2 shows our architecture, which is used by both our class-specific models and the base model. It consists of an encoder (blue module) and a decoder (green module). The encoder takes in input (e.g., a semantic and edge map in our base model, more details of the content of the input will be introduced in the section 3.2) and processes it into a latent code z and a spatial feature tensor $\phi\prime$. The $\phi\prime$ is next processed by the feature cropping model and fed into decoder together with z to synthesize realistic images. Next, we will introduce details of the encoder, the feature cropping, and the decoder.

In the encoder, we first have a several of conv and downsampling layers ($Encoder^1$) , then we extend its backbone at the last N layers (refer E^2_{BU} and E^2_{TD} in the Fig. 2) with a feature pyramid [22], enhancing higher resolution features (that are more accurately localized to the input) with lower resolution features (that are more semantic and has larger receptive field to capture global information). For example, if the input was a segmentation map then the higher resolution features would be more aligned with the input layout whereas the lower resolution features would have more global information about all the classes present in the segmentation map. As shown in the Fig. 2, the second part of the encoder, E^2, consists of two pathways: bottom-up (E^2_{BU}) for downsampling and top-down (E^2_{TD}) for upsampling and merging features from E^2_{BU}. The output of E^2_{TD} is $\phi\prime$. Note that z is the output of E^2_{BU} without being processed by E^2_{TD}.

Next, the spatial feature tensor $\phi\prime$ is fed into the feature cropping module, a fixed operation without learnable parameters. In class-specific models, the feature cropping is predefined to crop class instance regions from the $\phi\prime$ to get the starting feature tensor ϕ for the decoder. Refer the section 3.2 for more explanations. For the base generator, the feature cropping module is identity mapping; i.e., $\phi = \phi\prime$.

The green module in our architecture is the decoder adopted from StyleGAN2 [19, 32]. In StyleGAN2, the generator backbone takes a learnt constant as input. Instead, our generator takes the output from the feature cropping module as input since it provides the generator features that are relevant for the class at hand. As the resolution of ϕ is usually higher than the constant used in StyleGAN2 [19, 32], thus we skip its first K layers. More details of our architecture can be found in the supp.

3.2. Training pipeline

Base generator. As shown in the top left of Fig. 3, the base generator takes in segmentation map S and instance edge map E, to generate a base image I_b that covers the entire scene, i.e., $I_b = G_b(cat(S, E))$, where $cat(\cdot, \cdot)$ is a channel-wise concatenation. G_b is our base generator including both the encoder and the decoder architectures in Fig. 2. Using a spatial feature tensor as input to StyleGAN2 rather than a vector [32] gives the model strong guidance on how the generated spatial structure should look like. By sampling different z, our model can generate different results given the same segmentation map.

Class-specific generator. To further improve the quality of smaller object classes, we train multiple class-specific generators as shown in the top right of Fig. 3. If we generate each object instance without taking into account their context, the generated instances might look great on their own, but the final result after compositing multiple instances may look inharmonious due to inconsistent orientation, color, or lighting among different objects. For example, without context, our lamp-specific generator will generate lamps that do not look compatible with the lighting environment of the

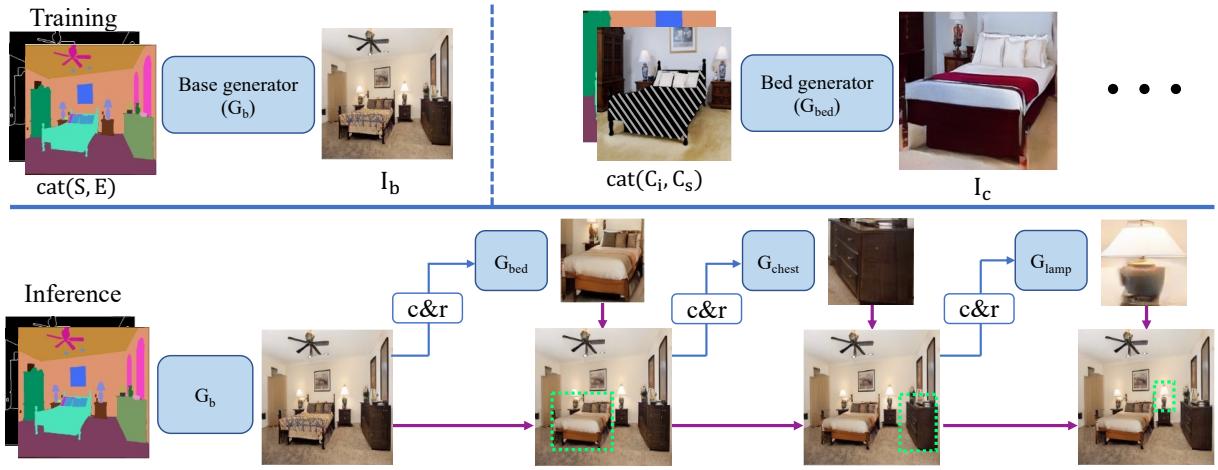

Figure 3. **Our Approach.** During training, our base generator is trained to generate an entire image, whereas our class-specific generators take in the cropped real image with object information removed (shaded region is filled with either zeros or low frequency information of the object) and cropped segmentation map as context information to generate the instance for the corresponding class. During inference, the base generator first generates the entire image and then the class-specific generators sequentially generate specific regions of the image while considering the previous generations as context. c&r means the cropping and instance information removing operation. Note that we do not show cropped semantic map as input of class-specific models in the inference pipeline for simplicity.

scene and shadows on the wall as shown in Fig. 6.

To address that, we provide some context information around the target instance as input to our class-specific generators. Specifically, we use the enlarged (2 times in both dimensions) box of an instance to crop both the real image I_{real_scene} and its segmentation map S to get the cropped real image C_i and the cropped segmentation map C_s for this instance. Next, we use two different ways to remove the instance information from the C_i before providing it for the generation (the shaded region in Fig. 3 top-right). The first one is to mask out the foreground region with zeros. The second approach is to blur the foreground region to retain only the low frequency information, in which case we hope the generated result will roughly follow the original instance color theme. The C_i (with instance information removed) and C_s are concatenated and used as context $C = cat(C_i, C_s)$ for the class-specific generator G_c to generate a specific instance $I_c = G_c(C)$. Note that C_i is not cropped from I_b, but from the real image during training. The reason is that the real image provides the perfect groundtruth to supervise the hallucination of the foreground object from the context while the generated I_b may contain artifacts. The feature cropping module (See Fig. 2) inside G_c is used to crop the spatial feature $\phi\prime$ to obtain the ϕ which is corresponding to the instance region only. The cropping box is calculate such that the final object is tightly generated within I_c. In this way, we can fully take advantage of decoder's capacity without generating any context outside the instance box.

To force the model to use C, we apply the perceptual loss [16] between the generated instance and the target instance, I_{real_ins}, which is directly cropped from the real image I_{real_scene} using the instance box without being enlarged. As the background pixels in I_{real_ins} already exist in C (i.e., C_i), the generator will autoencode the background region. In this way, the generator learns to gather hints from the surrounding context of the target instance and generates foreground pixels that look consistent with the background. Also, this loss can help our model maintains the low level frequency information provided in the C_i (the second case of removing instance information).

Training losses. We use the adversarial loss as well regularizations used in StyleGAN2 [32] and refer to all of them combined as $\mathcal{L}_{stylegan}$. For adversarial loss, the real distributions are $\{I_{real\_scene}\}$ and $\{I_{real\_ins}\}$ for our base and class-specific generator respectively. To regularize our encoder, we apply KL-Divergence [7] to the output of encoder z forcing it to follow normal distribution to support multimodal synthesis during inference, $\mathcal{L}_{kl}$. We also use perceptual loss [16]: $\mathcal{L}_{perceptual} = \sum_l ||V_l(I_{gen}) - V_l(I_{real})||_1$ where $V_l(\cdot)$ is output of l_{th} layer of a pretrained VGG network [28]. I_{gen} is I_b and I_c; I_{real} is I_{real_scene} and I_{real_ins} in our base and class-specific model, respectively. To summarize our overall training loss is: $\mathcal{L} = \mathcal{L}_{stylegan} + \lambda_1 * \mathcal{L}_{kl} + \lambda_2 * \mathcal{L}_{perceptual}$ The loss weights and the frequency of regularization within $\mathcal{L}_{stylegan}$ are the same as in StyleGAN2 [32]. More details can be found in the supp.

3.3. Inference pipeline

The bottom part of Fig. 3 describes our entire pipeline during inference. Starting from the input segmentation map

Scene	Classes	Training data sources
Bedroom	bed, chair, table	(1)+(4)
	chest, lamp, pillow	(1)+(4)+(5)
Human	shoes, face, upper clothes	(2)
Cityscapes	car	(3)+(6)
	person	(3)+(6)+(7)

Table 1. We can flexibly combine datasets to train our class-specific generators.

and instance edge map, we use our base model to generate the base image I_b. Then, the generated base image plays the role of providing context for each class-specific generator. Once an instance is generated by a class-specific generator, its result is composited into the base image. The new composited image becomes the new base image for the next instance. This process continues until all instances are generated. In practice, we generate instances based on their size (largest first) as larger objects are more likely to serve as context for smaller ones.

For better composition quality, we dilate and soften boundaries of both the generated instances and the instance masks before applying alpha blending. More details about the composition can be found in the supp. Though different from training where real images are used to provide context, we find that using the generated base image I_b to provide context at inference time also leads to good results.

4. Experiments

We perform quantitative and qualitative evaluations comparing our base model and CollageGAN model (combining results of class-specific generators) with prior arts.

Datasets. For baseline comparisons, we conduct experiments on the following datasets.
(1) **Bedroom**. We combine two existing datasets to get 74318 training images in total. We use the training set of $bedroom$ category from ADE20k [2]. Additionally, we use the $bedroom$ and $hotel\_room$ categories from places [38] and apply a segmentor [34] trained on ADE20K to get pseudo annotation. We use test set from the $bedroom$ category from ADE20K for evaluation.
(2) **Full human body dataset**. We collected 67560 high resolution images on full human body and annotated them with 24 common classes such as faces, upper-clothes, left shoes, right shoes. We randomly select 1/10 of the images as testing images (6757). We blur out the background region to focus the network capacity on generating the human instead of the complex background.
(3) **Cityscapes** [5]. It contains street scene images in German cities. The number of training and testing images are 3000 and 500, respectively.

We use all three datasets to train our base model and baselines. Our base model usually performs well for the classes of large extent in the scene, for example beds in the bedroom scene or large background categories like walls and floors. Hence, in order to show the impact of our CollageGAN, we train class-specific generators on classes of objects that are usually small and not synthesized well by our base model and baselines.

Datasets	SPADE	OASIS	LGGAN	Ours
Bedroom	45.84	39.13	NA	**34.41**
Human	38.53	8.65	NA	**7.51**
Cityscapes	59.68	50.90	61.46	**47.07**

Table 2. **FID (lower is better).** Our base model has consistently lower FID than the baselines.

Since we have separate models for different classes, we can use the following datasets as extra training data sources for generating bedrooms and cityscapes. For bedroom, we use (4) **iMaterialist** [6] and (5) **Indoor** dataset ($childs\_room$, $dining\_room$ and $living\_room$ from places dataset [38]). For cityscapes, we use (6) **Cityscapes_extra** [5] and (7) **Caltech Pedestrian** [26] dataset. More details about these additional data can be found in the supp. Table 1 summarizes the classes we choose and their training sources. We trained all the base models to generate 512×512 resolution images for bedroom and human dataset, and 1024×512 images for the cityscapes dataset. Since the resolution of each class varies, the class-specific generators are trained at 128×128 or 256×256 depending on average size of each class. For quantitative evaluation, We train all classes, except for person category in cityscapes, with blurred foreground region such that the model can try to maintain the color tone of instances in the base image during inference time. Our base model generates lower quality persons in cityscapes, hence blurred region wasn't very useful and we simply mask instead. More implementation details are in the supp.

4.1. Base model results

We compare our base model with SPADE [24] and its recent variants LGGAN [31] and OASIS [30] which are state-of-the-art for the segmentation map to image generation task. As we are targeting higher resolution than previous approaches (256×256 for ADE20K and 512×256 for cityscapes), we train their models at higher resolution and also provide them instance map for fair comparison. We trained them using their default parameters and verified the result quality of SPADE with the original authors. During training, all SPADE-based approaches are very memory demanding. For example, SPADE and OASIS takes around 16 GB per image to train 512×512 bedroom images while our model takes only 4 GB. LGGAN cannot fit a single image on a 32 GB V100 GPU for the bedroom dataset as it has a huge number of parameters due to the separate conv layers for each class. We are able to run LGGAN for the datasets with fewer classes like human and cityscapes (takes around 32 GB per image) but training is extremely slow; it would take more than a month for the human dataset with eight 32 GB V100 GPUs. Hence we only compare against LGGAN on cityscapes which is faster to train due to fewer images.

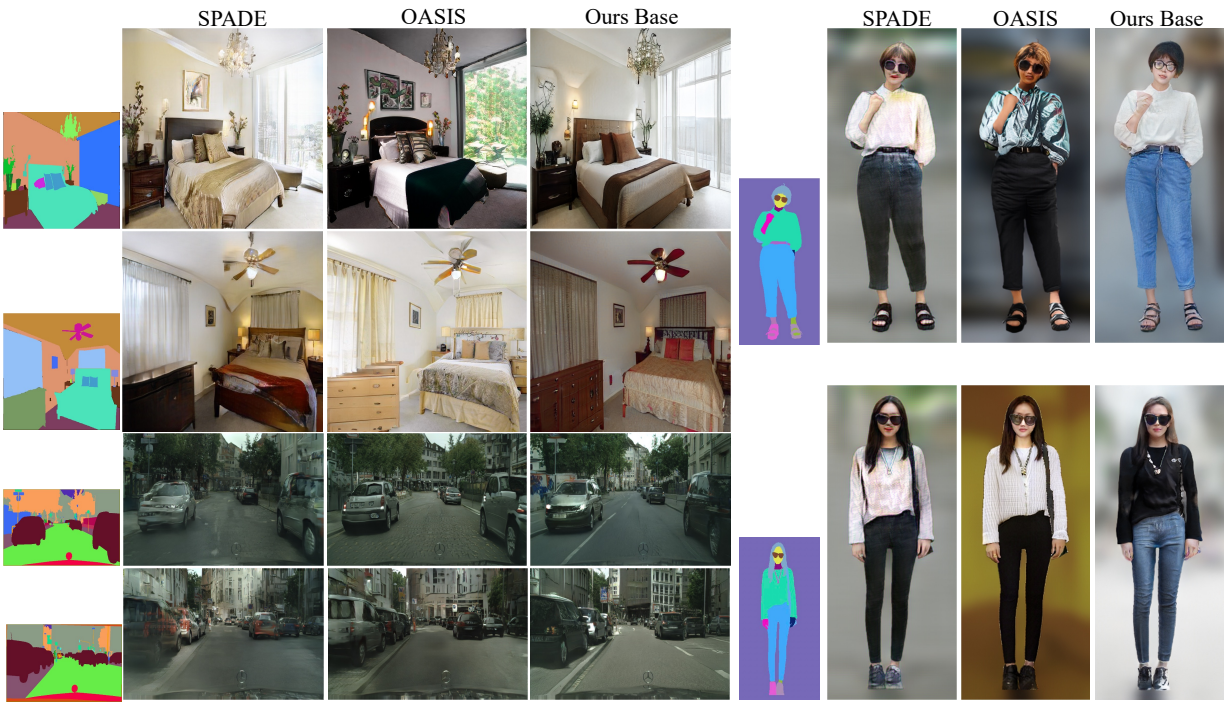

Figure 4. Visual comparison of segmentation map to image synthesis results on the ADE20K bedroom (512x512), full human body (512x512), and Cityscapes (1024x512) datasets. Our base model generates much more realistic images compared to SPADE and OASIS.

Datasets	Ours vs SPADE	Ours vs OASIS	Ours vs LGGAN
Bedroom	90.0%	73.2%	NA
Human	82.4%	63.2%	NA
Cityscapes	59.2%	35.2% (83.6%)	62.0%

Table 3. Percent of times users preferred our base model over baselines. Our base model is consistently preferred over baselines except for cityscapes when compared with OASIS. But our CollageGAN model is preferred 83.6% times over OASIS.

Quantitative results and human evaluation. In Table 2, we compare FID [12] of our base model with all the other approaches. Our base model gets the lowest FID on all three datasets.We also conduct user studies on Amazon Mechanical Turk (AMT). Specifically, we show the segmentation map and two generated images (ours vs one baseline) side-by-side to the AMT workers. We employ a two-alternative forced choice option for workers by forcing them to choose one image which looks more realistic. We showed 250 image pairs for each comparison and asked 5 unique workers to judge each pair. Table 3 shows our human evaluation results. Users strongly favored our approach over the baselines except for OASIS on the cityscapes dataset. The reason is that our base model sometimes generates less details for smaller objects when the training set is small (cityscapes has only has 3000 images). We can overcome this issue by using our CollageGAN consisting of class-specific generators for the smaller objects like car and person in the cityscape dataset. We then compared the results of our CollageGAN model against that of OASIS and found users preferred our approach 83.6% times. Please note that for a fair comparison we did not use any additional training data for the CollageGAN model in this case.

scenes	Bedroom					Cityscapes		Human	
classes	chest	chair	pillow	lamp	table	car	person	face	shoe
FID (base)	146.15	165.24	127.67	88.20	125.48	44.50	98.88	15.71	33.33
FID (CollageGAN)	132.12	155.52	136.79	80.12	119.44	30.42	82.34	13.54	29.87
User Favor	71%	70%	33%	62%	60%	94%	89%	84%	69%

Table 4. **Per class FID and user preference.** For most classes, our CollageGAN model is preferred over our base model.

	FID ↓			User study ↑	
	I: Base	II: w/o extra	III: w/ extra	I vs. II	I vs. III
car	44.50	36.71	30.42	23% / 77%	6% / 94%
person	98.88	88.47	82.34	13% / 87%	11% / 89%

Table 5. **Additional data ablation study.** I is our base model. II and III are CollageGAN models without and with extra data.

Qualitative results. In Figure 4, we compare our base model against SPADE and OASIS. Our generated images look more realistic and detailed. For example, the bed sheet in generated bedrooms contain more textures and clothes on generated humans contain more wrinkles. We also found OASIS to have some boundary artifacts which can be seen by zooming in on the OASIS generated human images. Our model can also generate multiple images corresponding to the same segmentation map by sampling different z. We show these results in the supp.

4.2. CollageGAN model results

Our CollageGAN model consisting of class-specific generators can further improve the quality of the images generated by our base model which we evaluate next.

Quantitative results and human evaluation. We first

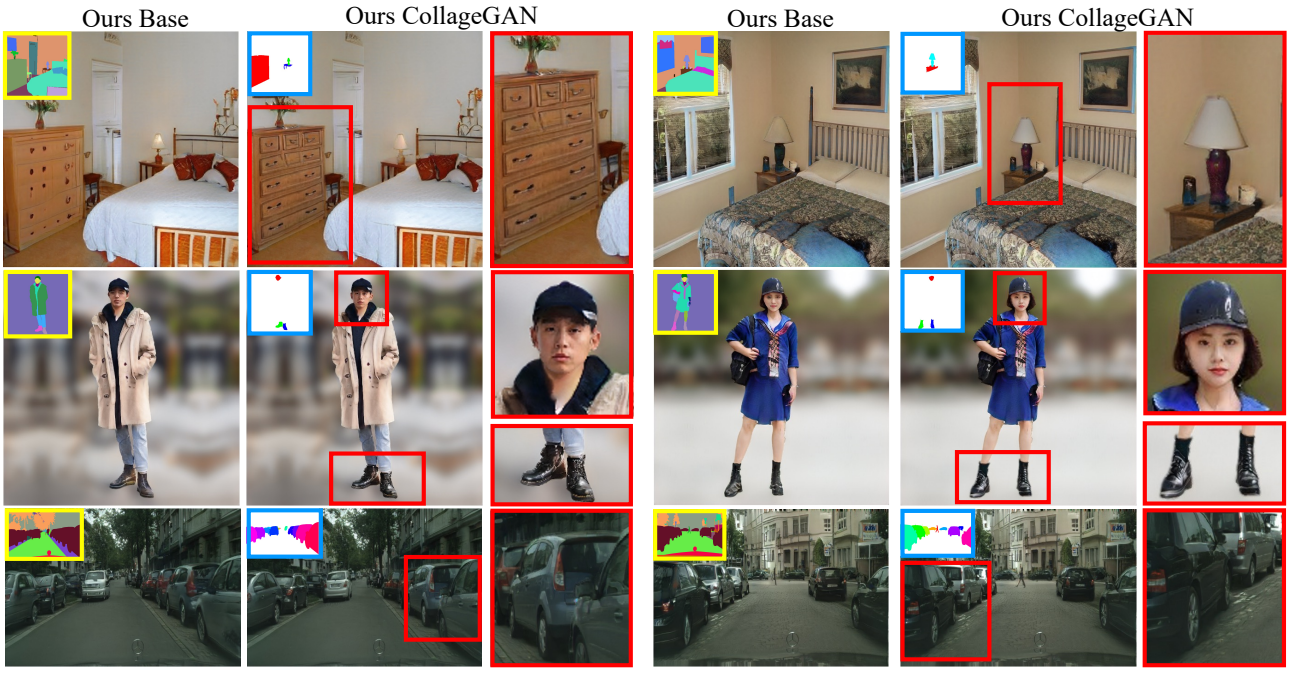

Figure 5. **Comparing our base model with our CollageGAN model.** Our CollageGAN model takes advantage of class-specific generators to provide more details to specific classes in the image generated by the base model.

compute per-class FID comparing our base model with our class-specific generators. Specifically, we crop each instance from both the original base image and the composition image generated using CollageGAN, and resize them to the average crop size over all instances in the class. We also conduct AMT evaluation, in which workers are shown class instances generated by our base model and by class-specific generator, and asked to choose the more realistic one. We show segmentation map at top with target instance highlighted, and two images at bottom (cropped from base and composition image). For each class, we show 100 pairs of images and 5 unique users are asked to judge each pair. In Table 4, we report per-class FID of our base model (first row) and our CollageGAN model (second row) and also the percentage of time users prefer our class-specific generator over the base model. Our class-specific generator consistently obtains lower FID and higher user preference in all object categories except for the pillow class. We observed that for the pillow class, the base model usually generates a pure white color pillow (which is the most common color in the training data) which is simple and usually looks realistic. Hence, our CollageGAN model does not add much value and any small boundary artifacts due to composition can become more noticeable.

Qualitative results. In Figure 5, we show results of compositing the pixels generated from our class-specific generator on top of the image generated by the base model. In the first column, we show the segmentation mask (yellow inset) and the corresponding image generated by our base generator. In the second column, we show our CollageGAN model results (highlighted class instance in blue inset show composited class instances). Finally, in the third column, we show the zoomed in view of the composited class instances which contain more details. For example, the chest in the first row contains more structural details like box boundaries and handles. The face region in the second row and the car in the last row look more realistic.

Ablation study. We first study the impact of training our class-specific generators with additional training data. Table 5 reports the result on cityscapes dataset (refer supp for the bedroom dataset), we can see that our class-specific generators perform better than the base model even without using any additional data (both in terms of user preference and FID). This shows the advantage of class-specific weights and centrally aligned data. Using additional training data further improves FID and user preference.

Next, we study the impact of providing context information C as input to our class-specific generators. In Fig. 6, our lamp generator trained with context generates lamps that are consistent with the surrounding lighting condition whereas the model trained without context fails to do so. For this ablation study, our lamp generator trained with context does not use blurred foreground during the training and inference time, so the network only relies on context to determine the lamp color. Similarly, the last two rows indicate that the model may fail to infer correct gender and skin color when trained without any context. We also observe that sometimes our class-specific generator trained with context

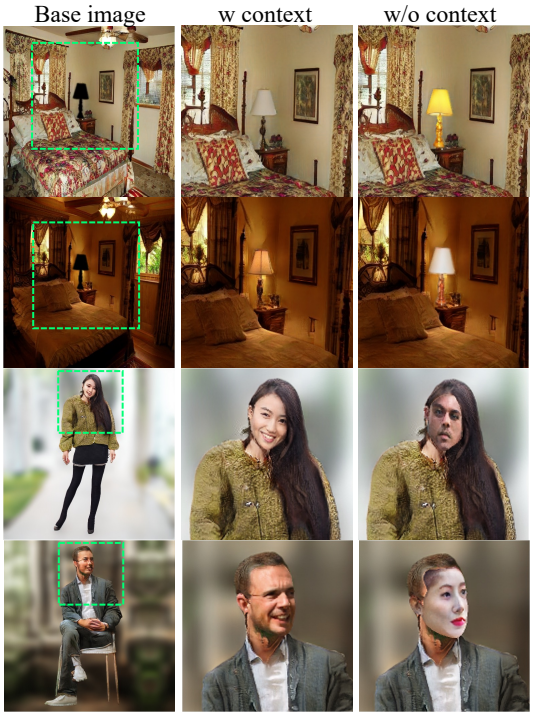

Figure 6. **Context analysis.** The last two columns show the importance of context during our class-specific model training.

Figure 7. **Replacing class instances in the real image**. Images in the red box are real images. Here bed and uppercloth generator is used to replace the original objects.

Figure 8. **Out-of-distribution generation.** Our car specific generator can generate cars on the sidewalk or on the middle of the road, which do not exist in the training distribution.

can correct the mistakes (like incorrect car orientations in Figure 5 bottom row) made by our base generator. Please see the supp for more results.

Applications. Apart from generating more detailed and realistic images, class-specific generators can be used for several interesting applications. In Fig. 7, we use our class-specific generators to replace a single object in the real image without affecting the other image parts. We mask out the object instance we want to replace and give the remaining real image as context for our class-specific generator. In Fig. 7, we replace the shirt in the top row and the bed in the bottom row. Our generated region looks harmonious with the rest of the image while being different from the original. Note that here we train new bed and upper clothes generators by masking out foreground region as we want the model to generate diverse textures rather than following the original color tone in the real image.

Both our base model and SPADE-based models are trained to generate the whole scene at once and as a result it is difficult for them to generate objects outside their original context. In contrast, our class-specific generator can achieve this as they are trained to only generate a fixed class. Fig. 8, shows that our class-specific generator can generate a car on sidewalks instead of road while SPADE fails. Finally, we can also use our class-specific generator to generate a very high resolution image (4096×4096) where important parts like face can be generated at high resolution while the remaining parts can still be generated at lower resolution by our base model. Please see supp for these results.

5. Discussion and Limitations

We introduced a conditional version of StyleGAN2 architecture as a powerful base model for high quality and high resolution semantic image generation. We used the idea of leveraging context-aware class-specific generators to further enhance the quality of results, especially for small foreground objects. Our base model generates more realistic images than baselines, but does not perfectly align with the input segmentation map. The reason is that we downsample the segmentation map to 32×32 resolution (feature tensor $\phi\prime$) before injecting it into the decoder. We argue that weak conditioning from the segmentation map could give our model more freedom to generate realistic pixels and make our results more robust to inaccurate segmentation maps due to human or algorithm mistakes. In the supp, we provide a detailed study to evaluate the alignment between the input segmentation map and the generated images for our base model and baseline approaches.

Acknowledgements. This work was supported in part by NSF CAREER IIS-1751206.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *ICML*, 2017.

[2] Xavier Puig Tete Xiao Sanja Fidler Adela Barriuso Antonio Torralba Bolei Zhou, Hang Zhao. Scene parsing through ade20k dataset. In *CVPR*, 2017.

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019.

[4] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NeurIPS*, 2016.

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[6] CVPR-FGVC5. https://www.kaggle.com/c/imaterialist-challenge-furniture-2018. 2018.

[7] Max Welling Diederik P Kingma. Auto-encoding variational bayes. In *ICLR*, 2014.

[8] Ian Goodfellow. NeurIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[10] Shuyang Gu, Jianmin Bao, Hao Yang, Dong Chen, Fang Wen, and Lu Yuan. Mask-guided portrait editing with conditional gans. In *CVPR*, 2019.

[11] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *NeurIPS*, 2017.

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Gunter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017.

[13] Tobias Hinz, Stefan Heinrich, and Stefan Wermter. Generating multiple objects at spatially distinct locations. In *ICLR*, 2019.

[14] Rui Huang, Shu Zhang, Tianyu Li, and Ran He. Beyond face rotation: Global and local perception gan for photorealistic and identity preserving frontal view synthesis. In *ICCV*, 2017.

[15] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

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

[17] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *CVPR*, 2018.

[18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *ICLR*, 2018.

[19] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[20] Kathleen M Lewis, Srivatsan Varadharajan, and Ira Kemelmacher-Shlizerman. Vogue: Try-on by stylegan interpolation optimization. *arXiv preprint arXiv:2101.02285*.

[21] Peipei Li, Yibo Hu, Qi Li, Ran He, and Zhenan Sun. Global and local consistent age generative adversarial networks. In *CVPR*, 2018.

[22] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[23] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. In *ICLR*, 2018.

[24] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019.

[25] Yipeng Qin Peter Wonka Peihao Zhu, Rameen Abdal. Sean: Image synthesis with semantic region-adaptive normalization. In *CVPR*, 2020.

[26] Bernt Schiele Pietro Perona Piotr Dollar, Christian Wojek. Pedestrian detection: A benchmark. In *CVPR*, 2009.

[27] Kripasindhu Sarkar, Vladislav Golyanik, Lingjie Liu, and Christian Theobalt. Style and pose control for image synthesis of humans from a single monocular view. In *ArXiv:2102.11263*, 2021.

[28] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2015.

[29] Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. FineGAN: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. In *CVPR*, 2019.

[30] Vadim Sushko, Edgar Schönfeld, Dan Zhang, Juergen Gall, Bernt Schiele, and Anna Khoreva. You only need adversarial supervision for semantic image synthesis. In *ICLR*, 2021.

[31] Hao Tang, Dan Xu, Yan Yan, Philip H. S. Torr, and Nicu Sebe. Local class-specific and global image-level generative adversarial networks for semantic-guided scene generation. In *CVPR*, 2020.

[32] Miika Aittala Janne Hellsten Jaakko Lehtinen Timo Aila Tero Karras, Samuli Laine. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020.

[33] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018.

[34] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[35] Jing Shao Xiaogang Wang Hongsheng Li Xihui Liu, Guojun Yin. Learning to predict layout-to-image conditional convolutions for semantic image synthesis. In *NeurIPS*, 2019.

[36] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. Lr-gan: Layered recursive generative adversarial networks for image generation. *ICLR*, 2017.

[37] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017.

[38] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *PAMI*, 2017.

[39] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *ICCV*, 2017.