Semantic Image Synthesis using a Spatially Adaptive feature-focused VAE-GAN with Switchable Normalization

Krishna Gorrepati
Stanford University
450 Serra Mall, Stanford, CA 94305
krishg@stanford.edu

Alec Lau
Stanford University
450 Serra Mall, Stanford, CA 94305
alau2@stanford.edu

Abstract

The ability to generate photorealistic images given a semantic layout is an interesting and active field of research. In semantic image synthesis, a semantic layout is processed through convolutions, normalization, and nonlinearity. Here we attempt to improve spatially-adaptive normalization layers (SPADE) as in [1]. We do this by applying switchable normalization layers as in [2], and creating a new generator architecture for SPADE inspired by photorealistic style transfer architectures in [7]. Spatially-adaptive normalization (SPADE) is preferable as it preserves semantic details that would otherwise be lost through normalization. By combining SPADE with switchable normalization, this architecture has adaptable normalization choices without loss of semantic information. By applying our new architecture, we find that our novel architecture with switchnorm produces more detailed images, has greater pixel accuracy, but is also less predictable than that of the baseline.

1. Introduction

Semantic Image Synthesis generates photo realistic images from semantic image masks. Each mask or segmentation map includes a pixel by pixel classification of the image into categories. This pixel level classification has been incredibly useful in image processing and style transfer. Generating images from these masks can be used in content generation from virtual environments to architectural designs.

Recent methods attempt to generate images by stacking convolutions between the input and output layers. However, we start to see issues with normalization washing away semantic mapping information due to the input layer being downsampled by the encoder and then later upsampled by the decoder to decrease convolution layers [1]. To prevent this network from losing information, we propose using spatially adaptive normalization (SPADE) which allows the segmentation map to control normalization parameters in each layer with switchable normalization. Since Spade preserves some unwanted noise, we hope switchable normalization's inclusion of channel wise norm, batchnorm, and layer norm will result in more realistic photo images.

Also, the interpolation layer in the generator of SPADE further compresses the segmentation map data into a very small tensors relative to the segmentation map size. We propose including downsampling layers inspired by [7], we can preserve more information from the map labels. By replacing the upsampling layers in the original SPADE with unpooling layers, we now have guided upsampling layers. We hope this new architecture coupled with Spade and with the addition of switch normalization will increase user preference for images.

We conduct experiments on the ADE20k which contains more than 20K scene-centric images annotated with objects and object parts. The dataset contains an image with a semantic ground truth shown side by side in Figure 1. This data is used as input to train the GAN and modified SPADE architecture. The conditional image synthesis in the case would be given the segmentation map on the right, generate an image like that of the left. [3]

Figure 1. Training image (left) with semantic ground truth (right) from ADE20K. Data was taken from MIT CSAIL[4]

2. Related Work

Deep generative models are used to synthesize photo-realistic images. Among the most popular deep generative models are general adversarial networks (GANs) and variational autoencoders (VAEs). VAEs consist of an encoder-decoder architecture as described in [9], where the input is fed through a neural network (called the "encoder") to encode the input, and the encoded input is decoded through other convolutional layers (the "decoder"), the intent being to teach the decoder a correct way to generate an output. As noted in [10], VAEs themselves tend to have trouble producing detailed photorealistic images. GANs, on the other hand, have two neural networks: a "generative" network and a "discriminative" network, where the generative network learns to create data intending to fool the discriminative network, which learns to tell the actual input data from the generated data. In [8], VAEs and GANs are proposed to work in tandem, by having the GAN descriminator be the decoder in the VAE.

Conditional image synthesis requires different input. In SPADE and in this work, we wish to generate images given a segmentation map. A segmentation map is an image where parts of the image are segmented into differing labeled categories. For instance, a landscape image's segmentation map might consist of three colors: one for the parts of the image with grass, one for mountains, and one for the sky.

Normalization layers reduce covariate shift in data. That is, introducing normalization layers help to preserve the distribution in data during training, thus allowing for larger learning rates and general easier trainings. There are three popular types of normalization: instance normalization (In), batch normalization (Bn) [5], and layer normalization (Ln) [6].

Switchable normalization (Sn), developed by [2], is a combination of channel-wise, batch-wise, and layer-wise normalization (In, Bn, and Ln, respectively). The idea behind Sn is a weighted sum of these three normalizations, given by the formula:

$$\hat{h}_{n,c,i,j} = \left(\frac{h_{n,c,i,j} - \sum_k w_k \mu_k}{\sqrt{\sum_k w'_k \sigma_k^2 + \epsilon}}\right)\gamma + \beta \qquad (1)$$

Here we have 6 learnable weights w_k, w'_k. Computationally this is implemented by taking advantage of the fact that we can calculate the statistics of batch normalization and layer normalization using instance normalization. This formulation allows an adaptable protocol for normalization for many different architectures, and is robust to changes in minibatch size.

Photorealistic style transfer involves changing the style of a photo to that of a reference photo. For example, one could create an image of a snowy Miami street by applying the style of a snowy landscape with an image of the Miami street one would want to see with snow. [11] formulates stylization as an image reconstruction problem. The whitening and coloring transform (WCT) in [11] is a VAE for general image reconstruction with the architecture in Figure 2. The idea behind this architecture is to directly

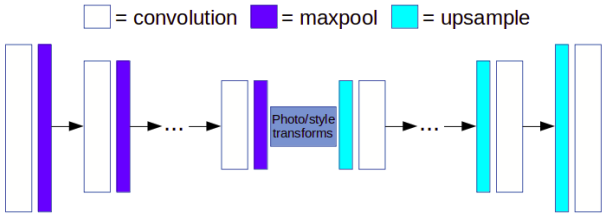

Figure 2. WCT architecture for generating images.

match feature correlations of the style image and the content image (i.e. the image we would like to stylize). This is done using the photo and style projections in the middle of the architecture. It uses the well-known VGG model from [13] to create a feature map that is then fed through a pre-trained decoder.

PhotoWCT is an architecture developed by [7] improving upon this architecture. The idea behind this adjustment is that maxpool layers remove information from the photo, and thus detail that reduces the photorealism of the result. This is illustrated in Figure 3.

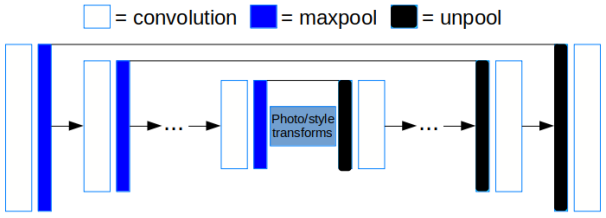

Figure 3. PhotoWCT architecture for generating images. Notice the upsampling layers in WCT are replaced by unpooling layers. The unpooling layers are used with maxpool masks, recording the indices of the maximum value in the maxpool region. [7]

Spatially adaptive normalization, introduced by [1], is a state-of-the-art architecture for semantic segmentation GANs. One of the difficulties of semantic image synthesis, noted in [1], is the loss of detail in the semantic layout of the input. This is due to normalization in the network; if an input is given a uniform segmentation map, instance normalization makes all post-normalized pixels identical due to the averaging through a uniform set. If the input were a uniform segmentation map with one label, or equal amounts of other labels, the normalization removes any distinction. Thus, the idea of spatially-adaptive normalization is to re-

update the model with the input. This is done with the architecture illustrated in Figure 4, henceforth referred to as the SPADE layer. This SPADE layer is implemented in a

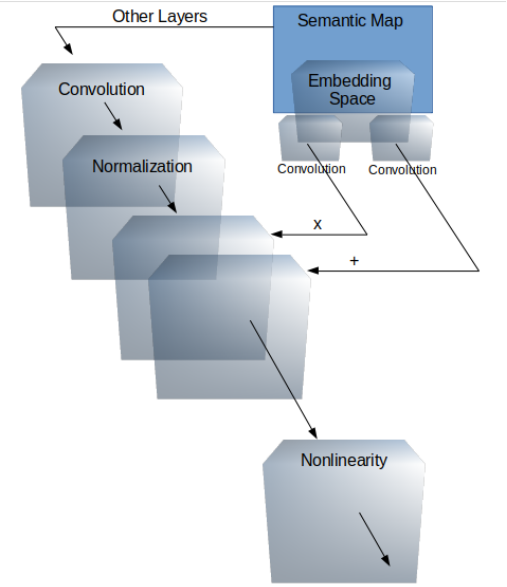

Figure 4. How the SPADE architecture "reminds" the normalization layer of semantic information.

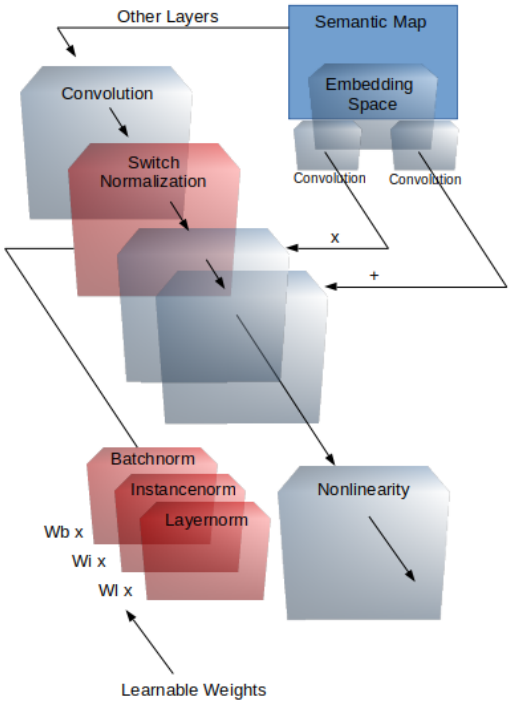

Figure 5. SPADE structure with learnable normalization.

"ResBik" layer consisting of SPADE, ReLU, a 3×3 convolution, SPADE, ReLU, and another 3×3 convolution. The architecture of the entire model is that of a generator with alternating ResBik layers and upsampling layers.

3. Methods

3.1. Implementing switchable normalization

We seek to make improvements to semantic image synthesis by first using learnable normalization via [2]'s switchable normalization program. By applying spatially-adaptive normalization as developed by [1] to this normalization layer we hope to optimize normalization in our model with minimal loss of semantic information. In our implementation, we ignored the sum over batch normalization. The Sn formulation allows for an adaptable normalization protocol for many different architectures, and is robust to changes in minibatch size. Indeed, the results in [2] show promising results when applied to instance segmentation in COCO, a segmentation dataset very similar to our ADE20K dataset, and ADE20K itself. It is this reason, coupled with the relative ease of implementation, that we selected adding Sn to SPADE as one of our attempts at improvement.

Our approach to optimize this SPADE model was to replace the instance of batch normalization in SPADE with Sn, as illustrated in Figure 5.

The idea behind implementing Sn in the first place was due to observing some of the ADE20K images. For example, some landscape images contained very rough-looking rocks with smooth sky. Learnable normalization could remove unwanted semantic information that is steadfastly preserved by SPADE. Thus, learnable normalization was predicted to be a safer way to preserve photorealism. After preliminary testing for two epochs, we found that Sn produced smoother texture in regions with smooth ground truths, seeming to validate our initial predictions. After testing, we found that this implementation of switchable normalization generated noticeably better results than the baseline (more on this in the results section). Thus, we decided to keep this implementation with our further changes to this model.

3.2. Novel SPADE generator

In light of [7] and [8], we seek to apply spatially adaptive normalization (with our switchable normalization) to a VAE-GAN architecture inspired by PhotoWCT. In short, we introduce a VAE architecture based on VGG networks from [13] with photorealistic feature preservation in mind from [7]. In the original SPADE architecture, the generator started with a heavily downsampled segmentation map input. The generator architecture of SPADE is shown in Figure 6.

Through interpolation we lose potentially valuable information, as the compressed input tensor is relatively small. Thus, we seek to preserve more information by downsampling layers. The architecture of the downsampling layers is

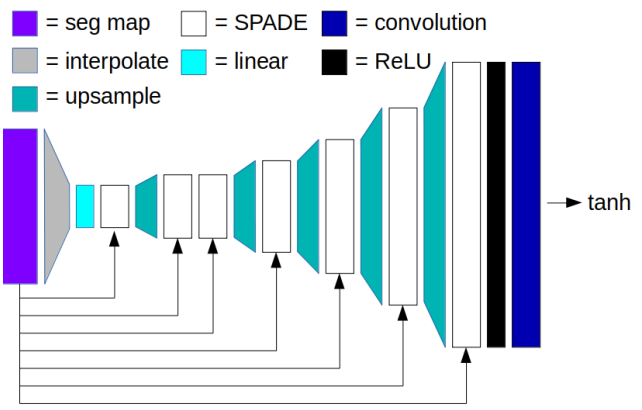

Figure 6. SPADE generator architecture

the same as that of [11], with a maxpool mask for later unpooling layers, as [11] reports success using this encoding network. By downsampling this way, and thus creating a VAE-GAN in the process, we have learnable ways to compress segmentation maps, a more computationally heavy but more intelligent way to downsample our map. In the original SPADE generator, resolution is increased by upsampling layers. However, resolution can also be increased by unpooling layers. We hypothesize that this architecture could be improved by increasing resolution in this way; unpooling layers are guided by information from the downsampling layers in our encoder. We hypothesize thus that, in addition to recovering information lost from maxpool layers in our encoder, the upsampling layers in our architecture are relatively blind methods of increasing resolution. The generator architecture we implemented to experiment with this idea is seen in Figure 7.

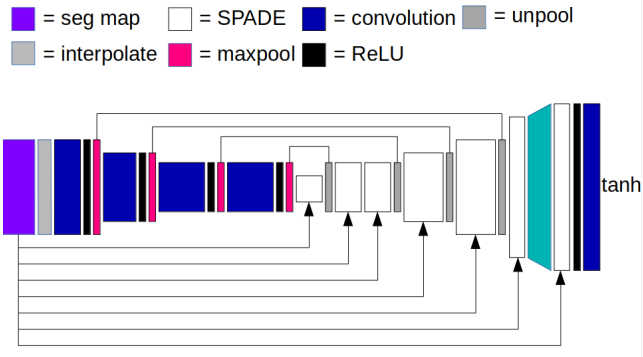

Figure 7. Generator with VAE inspired by [7] used with spatially adaptive normalization.

As one can see in Figure 7, we have replaced the plain upsampling layers in the original SPADE architecture with the unpooling layers, with the maxpool mask information given by our added downsampling layers. By creating learnable downsampling layers, we have implemented a VAE-GAN. Because this maxpool-unpooling relationship is reported by [7] better encode and decode photorealistic information, we hope this is a better generator, as more information about the segmentation map is included. One notices our architecture keeps the final upsampling layer, leading into the same final SPADE Resnet block (SPADE ResBlks), the same ReLU nonlinearity, and convolutional network. The $\tanh$ function is applied in our architecture as well. We included this last upsampling layer because, as noted in [1], cutting down on downsampling layers makes the original SPADE model less bulky. As such, we did not want to create a fully symmetric VAE as in [11] or [7], as training time is palpably slow for GANs with these datasets. We decided on allowing one original upsampling layer to remain, and observed that the training did not slow down noticeably with this implementation. It is for this reason that not every upsampling layer was removed from the original SPADE architecture. Another difference between our architecture and [7]'s architecture is the lack of pure convolutional layers after each corresponding unpool layer. This is because we felt the convolutions in each SPADE ResBlk layer following our unpooling layers would be able to accomplish anything the pure convolutions in Figure **??** might contribute to information flow.

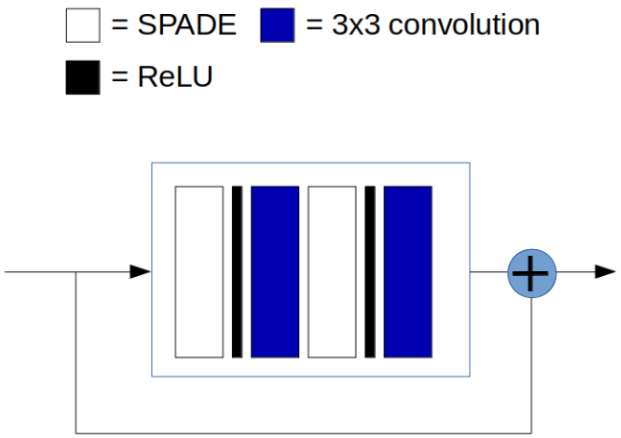

Figure 8. SPADE ResBlk architecture. We use the convolutions in this architecture as the plain convolutions in Figure 3.

The architecture above is the layer denoted by the white rectangles in Figure 6 and in Figure 7.

The above structure as the SPADE descriminator as the decoder for our VAE is inspired by the reports by [11] and [7] of its success. The base VAE structure is designed for style transfer: given an input content image I_C and style image I_S, the encoder architecture extracts each image's VGG features as in the encoder in [13], resulting in outputs $\epsilon(I_C)$ and $\epsilon(I_S)$. It then trains the symmetric decoder, then performs whitening and color feature transforms designed to

make the resulting output $WCT(I_C)$ satisfy the equation

$$WCT(I_C)(WCT(I_C))^T = \epsilon(I_S)(\epsilon(I_S))^T \quad (2)$$

Then the feature map $WCT(I_C)$ is fed into the decoder to obtain the stylized image.

The architecture of WCT was designed for artistic stylization. As expected, one runs into problems when stylizing a photo with the intent on creating a photorealistic output. The culprit, found by [7], is the removal of detailed information by the maxpool layers. This novel architecture illustrated in 3 reports promising results in creating photorealistic images. It is in this light that we created our novel generator architecture with the focus on features of the input, inspired by feature preserving VGG-based VAE architecture.

4. Data

We have trained and tested our models using the ADE20K dataset. ADE20K is a very popular dataset for training generative models for semantic segmentation. While there are other popular datasets with photos and semantic segmentation map ground truths, most notably the COCO dataset, ADE20K was most readily available from [4]. Upon seeing example images and semantic segmentation ground truths from both COCO and ADE20K, the difference between the two was deemed negligible. The ADE20K dataset contains 20210 training images along with 2000 validation images. For each image there is: an RGB image file, an object segmentation mask containing object class segmentation masks with the first two channels corresponding to object class masks, and the third corresponding to instance class masks, a parts segmentation mask, where the segmentation masks correspond to parts of the objects, and a .txt file providing a description of the image. There are 150 semantic classes in the ADE20K dataset. The resolution of each image is 1024×1024. See Figure 1 for an explicit example.

ADE20K is standard amongst image segmentation and contains "challenging scenes." [1] For this reason, and the fact that it is widely used in many papers on semantic segmentation, we chose not to do any data preprocessing. While data pre-processing would have induced much shorter training times, we are aiming at preserving photorealistic features through our experimental architecture, and preprocessing this data would have made the results of our experiment less clear, as this would have removed photorealistic details we would otherwise have liked to have tested our architecture on.

In describing the dataset, a picture is truly worth a thousand words, and millions of parameters. In viewing Figure 1, one sees a validation photo on the left with its semantic ground truth on the right. The semantic ground truth's different colors are labeled as one of the 150 semantic classes specified in ADE20K. Given a semantic layout like the semantic ground truth, SPADE, along with architectures like it, can create a photorealistic image derived from the training data.

We trained and tested our models on the unprocessed 1024×1024 resolution images, with a batch size of 16 for 20 epochs in our baseline and experiments, with 20000 iterations per epoch.

5. Results & Discussion

First we trained the SPADE model as given in [1] and ran a validation as a baseline. Training was with a batch size of 16 on 4 Nvidia Tesla P100 gpus. This was the maximal amount of gpus we would obtain from gcloud, and 16 was the maximal batch size we could use on our 4 gpus. We trained for 20 epochs with 20000 iterations per epoch. This is much lower than SPADE's 50 epoch training reported, due to constraints on time and money, and the large size of the dataset. All other parameters were set as default according to SPADE's documentation. This is because we wanted to create the optimal training model, and assumed that the authors of [1] had done ample hyperparameter tuning. For this reason, and in the interest of keeping our controls held constant, we did not modify any of the hyperparameters in the given SPADE. Then we trained our switchnorm implementation and ran the same validation, following with our novel generator. Here in Figure 9 we have resulting images generated by the baseline and our experiment. The first column represents a sampling of the validation photograph ground truths. The second column represents the validation from the baseline. The third represents the validation from our switchnorm implementation, and the fourth is from our novel architecture.

As one can see in Figure 9, after a relatively small number of training epochs, our switchnorm experiment gives more diverse and in most cases more realistic photorealistic details. This is notably not the case for all images. In a minority of validation images, we have bright, discolored "spotting" reported by [1] when testing the more common pix2pix architecture in [12]. In general this can qualitatively be described by increased diversity in features. For example, in the top row landscape image we used as an example validation image, there is more grass on the ground in our implementation than in the baseline. However, the "spottiness" is a downside of such feature diversity. This is likely because of the nonzero effect the instance norm and layer norm have on the model, compared to the baseline's simple batch norm. We conclude that a learnable, nonzero amount of batch normalization, layer normalization, and instance normalization provide a riskier but more vivid image. In the future a hard-coding of batchnorm would be desirable, as the bright spots would likely be normalized out by the more realistic pixels in the rest of the batch.

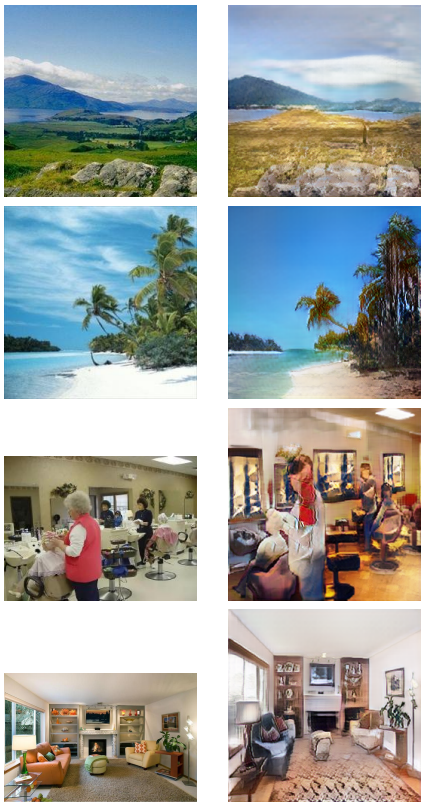

Figure 9. Validation images. The left column has the validation images ground truth photograph. The second column has the images generated our baseline

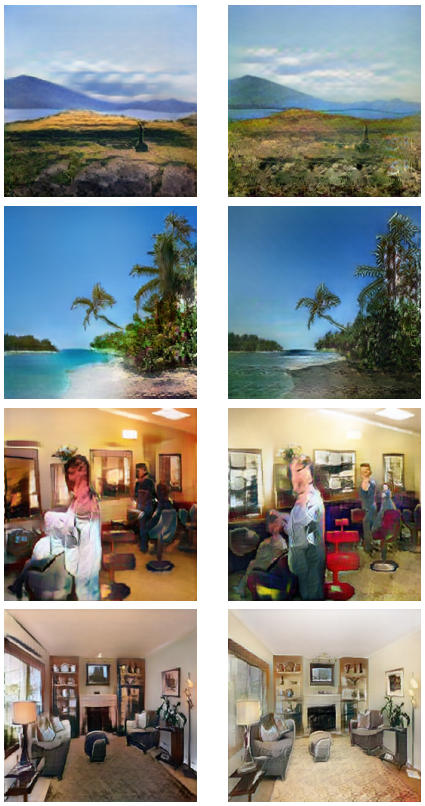

Figure 10. Images generated by SPADE just with switchnorm are on the column on the left. Images generated by our novel architecture with switchnorm on the right column.

We trained our novel SPADE generator architecture using the same parameters stated above. Here in Figure we have resulting images generated by the baseline and our experiment. The first column represents a sampling of the validation images. The second represents the validation from the baseline. The third represents the validation from our implementation. As one can see in Figure 9, after a relatively small number of training epochs, our experiment gives significantly more detail than the baseline or the switchnorm. However, there are the same problems run into by switchnorm, namely the bright and discolored spots. These are features in themselves, and thus are preserved by our novel architecture. This leads us to the conclusion that our implementation of switchnorm, or lack of a stronger batchnorm, is the cause for occasionally chaotic image synthesis. That is, the model is too feature-focused to do away with any outliers caused by lack of segmentation knowledge provided by the SPADE architecture. In our planning of the model, this did come up as a potential hazard, but it was decided that the SPADE ResBlk layers would contribute in an interesting way to counteract this. After looking though different environments, we can qualitatively conclude that our novel architecture outperforms the other two on landscapes, but is less realistic with interiors and faces. This may be because the amount of large collections of features in the latter environments confused the hyper feature-focused generator architecture, causing confusion of the GAN.

We recorded the losses from the baseline and two experiments at various iterations in Figure 12. These are the losses for the discriminator in detecting fake images (D_fake), the loss for the discriminator in detecting real images (D_real), the loss for the GAN (GAN), and GAN features (GAN_feat). Through each training model, we see the discriminator over time gets better at recognizing fake images and worse at recognizing real images. The generator continues to produce varied images to trick the generator. With our limited run of 20 epochs, all models roughly converged to a loss of .8 for the generator and discriminator.

6. Conclusions

Using spatially adaptive normalization to preserve semantic information in GANs is a promising technique for semantic image synthesis. By applying switchable normalization, we found that switchable normalization increases diversity in features, often leading to a more photorealistic image, but still occasionally leading to anomalies. We conclude that layer normalization and instance normalization

Figure 11. Losses for our baseline and experiments as recorded by SPADE. The first column is our baseline. The second columns is our switchnorm with original SPADE architecture. Graphed using Tensorboard

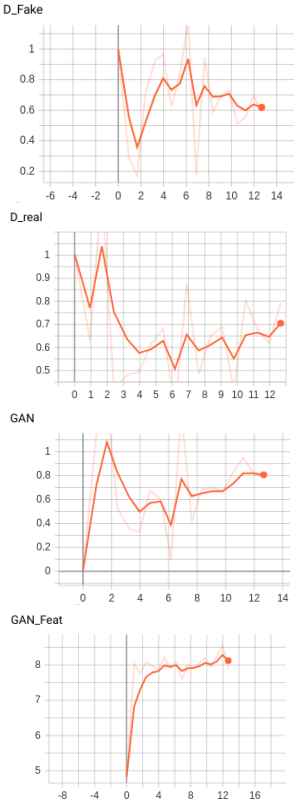

Figure 12. Losses for our novel architecture with switchnorm. Graphed using Tensorboard

contribute nontrivially to variance of generated features.

We also applied SPADE to our novel VAE-GAN architecture: the VAE being a VGG modified with an eye to photorealistic style transforms, we found that details were better generated than either the baseline or switchnorm. This is as we predicted; the feature-focused photo stylization centered VGG VAE-GAN was indeed feature focues. The smoothness of simple switchnorm becomes more obvious in comparison to our new architecture. This leaves the experimentation to whether switchnorm is actually optimal or not.

7. Future Work

In the future, we would first and foremost want to train our models using the full 50 epochs as per the documentation in [1], as we would like to see the completed model, and then most likely the photorealistic images seen in [1]. We would also like to hard-code a batch normalization layer into our switchnorm implementation layer, as normalizing over the batch of data would likely reduce the brightness or glaring colors of the spots discussed in the previous section. We would also like to experiment with different amounts of unpooling levels in our decoder, and see how this corresponds to photo detail being generated. Finally, we would like to run this novel architecture without switchnorm and see if the occasional chaotic image still occurs.

8. Contributions & Acknowledgements

A. Lau configured SPADE and implemented the novel architecture VAE-GAN architecture. K. Gorrepati implemented switchnorm, and calculated metrics through tensorboard.

As base code, the authors used SPADE as found here: https://github.com/nvlabs/spade/. The authors also used the switchnorm code found here: https://github.com/switchablenorms/Switchable-Normalization. To record results, tensorboard and and ADE20K evaluation script were used as found here: https://gist.github.com/zhanghang1989/e9370ed95bcc1bddcc7a407739d622d1

The authors would like to thank Woodrow Wang and Tyler Yep for invaluable advice on configuring tensorboard, as well as Peter Dun for useful advice for downloading research code. The authors would also very much like to thank CS231n TA Boxiao Pan for helping elucidate that

an original project idea was unfeasible, as he discovered that the base code was pushed to Github mid-debugging session.

References

[1] T. Park, M.Y. Liu, T.C. Wang, and J.Y. Zhu. Semantic image synthesis with spatially-adaptive normalization. *arXiv preprint arXiv:1903.07291,* 2019. 1, 2, 3, 4, 5, 7

[2] P. Luo, J. Ren, Z. Peng, R. Zhang, and J. Li. Differentiable learning-to-normalize via switchable normalization. In *International Conference on Learning Representations (ICLR),* 2019. 1, 2, 3

[3] Liu, Xiaolong, Zhidong Deng, and Yuhan Yang. Recent progress in semantic image segmentation. *Artificial Intelligence Review* (2018): 1-18 1

[4] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso and A. Torralba. Scene Parsing through ADE20K Dataset. *Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 5

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *In International Conference on Machine Learning (ICML)*, 2015. 2

[6] J. L. Ba, J. R. Kiros, G. Hinton. Layer normalization. *arXive preprint arXiv:1607.06450*, 2016. 2

[7] Y. Li, M. Y. Liu, X. Li, M. H. Yang, J. Kautz. A Closed-form Solution to Photorealistic Image Stylization. *European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 3, 4, 5

[8] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, O. Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2016. 2, 3

[9] D. P. Kingma, M. Welling. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[10] A. Dosovitskiy, T. Brox. Generating Images with Perceptual Similarity Metrics based on Deep Networks. *arXiv preprint arXiv:1602.02644*, 2016. 2

[11] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, M. H. Yang. Universal Style Transfer via Feature Transforms. *Neural Information Processing Systems (NIPS)*, 2017. 2, 4

[12] P. Isola, J. Y. Zhu, T. Zhou, A. A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *arXiv preprint arXiv:1611.07004*, 2018. 5

[13] K. Simonyan, A, Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICLR)*, 2015. 2, 3, 4

[14] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer. Automatic differentiation in PyTorch. *Neural Information Processing Systems (NIPS)*, 2017.