Image transformation with multi-artistic styles

Yifan Mei **Chenlai Shi** **Han Xiao** **Lijing Yang** **Miao Yang**

Abstract

This article is to combine 2 styles onto a selected content image. We develop our work based on the method proposed by Gatys et al. (2015). To be specific, we use selected layers in the pre-trained 16 layers of VGG Network on the ImageNet dataset to train our content image, and style images respectively. Later, we calculate each given layer's loss function of content image and style images. We also add a regularization which is calculated by total variation to balance the output graph's pixels for a better visualization. Then, we sum the loss functions and regularization up by assigning different weights to them. The first type of outputs are the texture synthesis, which corresponds to the extraction of each style. By the texture synthesis created, we find out that the method proposed by Gatys et al. (2015) only work well on colorful types of images. If the contradictions of pixels are not large enough in a style graph, the exaction of style can not achieve a good result. This finding is shown clearly after we combine each style with the content features which are extracted from content image where the style lacking of contradiction is hard to distinguish on the combination output graph. In the end, we output our final combination with 2 styles together on a content feature "canvas" by selecting the styles which are colorful by pursuing better visualization.

1 Introduction

Paintings and photos are shining artistic works. To reveal the aesthetic perspective of creators, professionals are necessary for analysis. However, for people without related knowledge in arts, tracking and using styles from artistic images is difficult. Thanks to the development of machine learning, artificial neural networks can currently trace and detect underlying styles of images. The detected styles can then be applied onto other images to inspire the creativity of normal people. In this project, an efficient method is applied to transform an image with a combination of two styles from other images, while the content of original image remains. Convolutional neural networks (CNN) and image transformation methods are applied.

CNN is an efficient model used to achieve reasonable performance on visual recognition tasks, and has been developed and progressed with successive models, i.e. LeNet, AlexNet, GoogLeNet, VGGNet and ResNet. Similar to ordinary Neural Networks, CNN are made up of neurons that have learnable weights and biases, which takes raw image pixels as input and output class scores at the other end. CNN consist of a sequence of layers including CONV layer for weights computation, ReLU layer for elementwise activation, POOL layer for downsampling operation and FC (fully-connected) layer for class score computation. Three main hyperparameters are generally used to control the size of output volume for each layer: the depth of CONV layer, a.k.a. the number of filters in a column that look at the same region of input, the stride with which the filter is slid pixel-wise, and the size of zero-padding around the border of input volume. From the perspective of memory and computational time cost, the largest bottleneck of CNN construction would be the memory bottleneck. Efforts have been made to promote more efficient CNN implementations, such as Parameter Sharing scheme to reduce the number of parameters, running CNN only at test time which mitigates the memory load by storing only current activations at any layer.

CS760 FALL 2017 Final Project.

Image transformation reconstructs input images into different output images. The transformation denotes a set of techniques both in the category of image processing and computer vision. It can be used to upgrade the quality of images by getting rid of noise, colorizing the original grayscale, and increasing the resolution of images. One highlighted application of image transformation focuses on the extraction of semantic information from the scene. As an artistic perspective of image transformation, style transfer has been introduced as a way of reconstructing images to the style of other images. Features of content in content image are recomposed by synthesizing the textures from style image. Gatys et al. (2015) discovered that the style information and content information can be separated in CNN. Based on this statement, image stylization can be accomplished by combining extracted content info and style info respectively from different images. Gatys et al. (2015) used a pretrained VGG-16 network (Simonyan and Zisserman, 2014) to get information from images. Input of the transformation were an empty image, a style image and content image. The empty image was transformed to synthesize the content info in content image and the style in the style image.

This project specifically take advantages of the beneficial sides of CNN architectures in image stylization. Two styles, instead of usual one style, are mixed to transfer the content image into a different artistic work. Works done by Gatys et al. (2015) are studied. However, the traditional Chinese ink art was not able to provide accurate artistic style by the method.

In this report, literature review, the methodology and outputs of the method are followed in sequence. The limits of the methodology and possible future works are then discussed at the end of this report.

2 Literature Review

Image stylization requires content and style extraction from images. In this section, related works of content reconstruction are presented first. Reviews of texture synthesis are then stated, followed by recent works of image stylizations.

2.1 Content Reconstruction

This project applies CNN as the neural network architecture to gain content information from images for further image stylization. The information in each layer of CNN should be understood first to know how to extract content from images.

Many efforts have been done to understand and visualize the information of CNN outputs in each layer. Krizhevsky et al. (2012) proposed using the feature activations induced by an image at the last layer of CNN as a representation of image information. If two images produced feature activation vectors with a small Euclidean separation at the final layer, the two images were considered similar. Van der Maaten and Hinton (2008) presented t-SNE method to handle the dimension reduction of images. The t-SNE technique reduced the multi-dimensional image data into 2 dimensions, and the visualization and categorizing of images were easier. This technique was also applied into clustering graphs by activations from the last layer of a deep CNN (Karpathy, 2016) [1]. Yosinski et al. (2015) introduced 2 methods to display what information was gained in each layer of CNN. The first method simply displayed the activations of each layer to give intuitions of input images. The second method used gradient ascent to provide regularized optimization. A stochastic random image was optimized into pretrained CNN, and the optimization tried to reduce difference between one specific layer activations of the random graph and activations at the same layer from another image. With regularizers increasing the degree of visualization, the final output image showed what information was captured by the CNN layer. Springenberg et al. (2014) proposed a way to visualize the content of a filter. With running many images into the trained CNN, the activations of a specific filter were recorded. Image patches that corresponded to maximal activations were considered as visualization of the filter. Zeiler and Fergus (2014) did occlusion experiments to understand the importance of image part in classification. Part of the image was masked before feeding to CNN, and the probability heatmap at each mask location was drawn.

[1] Karpathy, A. (2016). http://cs.stanford.edu/people/karpathy/cnnembed/http://cs.stanford.edu/people/karpathy/cnnembed/

This project utilizes the gradient ascent method to extract content information from content images. As an optimization algorithm, gradient ascent is easy to calculate and has a strong theoretical foundation. It has been widely used in image visualization in recent years (Springenberg et al. ,2014; Simonyan et al., 2013). Details of the content extraction with gradient ascent will be discussed in the methodology section.

2.2 Texture Synthesis

Texture synthesis is originally a process of generating a new graph with repeating textures from sample images. Based on researches on texture synthesis, the stylistic information of a graph can be extracted. The extracted style can then be applied to other graphs for new art works.

Several texture synthesis approaches have been developed. Efros and Leung et al. (1999) developed an approach that generates pixels one at a time in a scanline order, then forms neighborhood of already generated pixels and copy nearest neighbor from input. Gatys et al. (2015) introduced a parametric texture model based on CNN. Their texture model using the hierarchical image representation in a deep convolutional network that was trained on object recognition in natural images. They generated a texture by extracting features of different size from the source image, and then compute a spatial summary statistic on the feature response. The new image with the same stationary description was generated by performing gradient descent on a random image that has been initialized with white noise. Ustyuzhaninov et al. (2016) demonstrated a new parametric texture model based on shallow convolutional network with random filter. Their model was able to qualitatively capture the perceptual differences between natural textures and produces high quality results. Jetchev et al (2016). reported a new data-driven texture synthesis method based on generative adversarial networks. The spatial GAN was created to achieve excellent performance of texture synthesis by extending the input noise distribution space from a single vector to a whole spatial tensor. Recently, Li et al. (2017). showed an improved texture synthesis approach based on a deep generative feed-forward network which can synthesize diverse results for each texture. They introduced the diversity loss and proposed an incremental learning scheme to train a deep network for multi-texture synthesis. The model produced high quality results and served as multi-style transfer model for image stylization.

In this project, the method of Gatys et al. (2015) is used to extract style information. Gram matrix was used as a method to estimate covariance and correlation among patterns through the graph. Details of the method is shown in the methodology section.

2.3 Texture Stylization

Contributions of content reconstruction and texture synthesis boosted the ideas of image stylization in 2015. By combining different content and styles, unique artistic graphs can be generated.

Gatys et al. (2015) introduced an algorithm using convolutional neural network (CNN) to separate and recombine content and style from arbitrary images. Content of an image was represented by feature response at high layers of CNN. The style of an image was shown by the Gram matrix in feature space, which consists of the correlations between the different filter responses over the spatial extent of the feature maps. The stylization algorithm was based on similarizing layer outputs of the content image and new image in one layer of CNN, and getting a similar Gram matrix of the style image and the new image on certain layers of CNN. The loss function used in the image stylization algorithm was a weighted combination of loss function of content reconstruction and texture generation. Because style representation and content representation were separable in CNN, emphasis on reconstructing content or style could be regularized by tuning the constants α and β in the loss function. Pretrained VGG network was used as the CNN structure. In Gatys et al.'s (2015) examples, output of the 'conv4-2' layer was matched as the content representation.

Johnson et al. (2016) combined the per-pixel feed forward neural network and the perceptual image stylization method of Gatys et al. (2015) to accomplish the image style transformation. Style transformation by the method of Gatys et al. (2015) generated training examples, and the training images were then applied into per-pixel supervised learning of a feed-forward neural

network. The results from Johnson et al. (2016) were similar with the ones by Gatys et al. (2015) both qualitatively and numerically in loss function value. But the method of Johnson et al. (2016) is three orders faster than Gatys et al.'s (2015) method with the supervised NN.

Image stylization method by Gatys et al. (2015) introduced distortions to photorealistic images. As a global average representation of textures, Gram matrix was not able to give faithful sematic accuracy. Luan et al. (2017) introduced a regularization to error function of the method by Gatys et al. (2017). The regularization term helped to constrain the reconstructed image to be represented by locally affine color transformations of the input to prevent distortions. Semantic segmentation was also applied into input and reference images to generate labels. Only the Gram matrix of the input image at specific segmentation channels would be updated.

3 Methodology and Implement Details

The main goal for our project is to combine 2 styles in a selected content image. Here, we choose the famous painting, Venice gondola at night, as our basic content image. Considering the lack of use of Chinese style paintings in previous related work, we choose 3 images from 3 Chinese painting styles: Traditional Chinese Painting, Chinese PaperCut, Chinese Ink Painting. To achieve our goal, based on the work of Hnarayanan[2], and Gatys et al. (2015), we applied the pre-trained 16 layer VGG network in the ImageNet dataset to all our images separately by using TensorFlow. Then, for the content image, we use all layers' results to calculate the loss function of content, while for each style images, we use each layers to build a corresponding Gram Matrix, and combine the Gram Matrixes together to produce the loss function for the style image. In the end, we combine the loss functions of one content and two styles together and output the graphs after the mixture. Beyond the general steps, we also output the graphs combining one content with one style, and the graphs only having style reconstruction to show more evidence about our conclusion about style selecting. More details are illustrated in the following.

3.1 Image Preprocessing

Preparation before plugging the indexes into the CNN model was to read the images into the TensorFlow by using a package named: Pillow. After such transformations, each image was represented by 3 channels: Red, Blue, and Green. Then, the images of both content and style parts needed to be resized so that they were all in the same size: 512 * 512 for height * width. In this way, corresponding to the 3 channels, each image can be represented by a 3 dimensional matrix whose size was 512 * 512*3. To concatenate the representations of content and style(s) into one graph, one more dimension was added to each image's matrix, and each matrix changed to size 1 * 512 * 152 * 3. Then, by subtracting the mean value of each channel of RGB calculated by ImageNet training set respectively, and reordering the RGB matrix to the order GBR to follow the order in Gatys paper, the first step, image pre-processing was finished.

3.2 Pre-trained Network

After the images had been preprocessed, the results can be plug into generator networks separately based on the VGG-Network[1]. VGG-Network is a kind of Convolutional Neural Network which focuses on the large-scale image recognition setting and has 16–19 weight layers. It was published by Karen Simonyan & Andrew Zisserman in 2015 as a conference paper at ICLR. Here, in the paper, VGG-16 is the selected network. Following Gatys, although in that paper, VGG19 was used, the fully connected layers and the final softmax classifier is not necessary for the training, thus, only the following layers was used: 'block1_conv1', 'block1_conv2', 'block1_pool', 'block2_conv1', 'block2_conv2', 'block2_pool', 'block3_conv1', 'block3_conv2', 'block3_conv3', 'block3_pool', 'block4_conv1', 'block4_conv2', 'block4_conv3', 'block4_pool', 'block5_conv1', 'block5_conv2', 'block5_conv3', 'block5_pool'.

[2]Harish Narayanan, https://github.com/hnarayanan/stylist/tree/master/algorithms

Then, to combine content and style(s) together, reconstruction of content, and each style were needed. To do so, the loss functions of these parts were calculated separately by using each layer's output from the corresponding VGG16 network. The following two parts show the process of reconstructing content and style precisely.

3.3 Content Reconstruction

To preserve our input images' structure while still not let the output image be exactly the same as the input image, we calculate the loss function of content. The content loss, Lcontents, is the Euclidean distance between feature representations of the content. For each layer l, the corresponding loss function of content is:

$$L^l_{content} = \frac{1}{2 \times F_l \times H_l \times W_l} \times \sum_{i,j} ||A^l_{i,j}(OC) - A^l_{i,j}(IC)||^2 \tag{1}$$

In this equation, *l* is the layer in the trained VGG16 network; $L^l_{content}$ is the loss function of content in layer; *i* is the i^{th} filter in layer; *j* is the position in the layer; A^l_{ij} is the activation of the i^{th} filter at position j in layer; *OC* is the output of content; *IC* is the input of content; F_l is the number of filters in layer; H_l is the height of the layer; W_l is the width of the layer.

3.4 Texture Synthesis

For the style loss, we first define a Gram matrix for each layer, which is used to compute the correlations between the different filter responses in any given layer of the corresponding VGG16 network trained for the given style image:

$$G^l_{i,j} = \frac{1}{F_l \times H_l \times W_l} \times A^l_{i,j}(A^l_{i,j})^T \tag{2}$$

In this equation, *l* is the layer in the trained VGG16 network; *i* is the ith filter in layer; *j* is the position in the layer; $G^l_{i,j}$ is the Gram Matrix in the i^{th} filter and position j for the layer; $A^l_{i,j}$ is the activation of in the i^{th} filter and position j in layer; F_l is the number of filters in layer; H_l is the height of the layer; W_l is the width of the layer.

The Gram matrix use each layer's information in the VGG16 network for a style image, and transform the three-dimensional matrix with size into a two-dimensional matrix with size . In this way, the complicated features in the style image preserved much better than using Method KNN.

By only capturing these aggregate statistics across the image, they are blind to the specific arrangement of objects inside the image. This is what allows them to capture information about style independent of content. The style loss is then the (scaled, squared) Frobenius norm of the difference between the Gram matrices of the style and combination images:

$$L^l_{style} = \frac{1}{2 \times (F_l)^2} \times \sum_{i,j} ||G^l_{i,j}(OC) - G^l_{i,j}(IC)||^2 \tag{3}$$

In this equation, *l* is the layer in the trained VGG16 network; L^l_{style} is the loss function of style in layer; *i* is the i^{th} filter in layer; *j* is the position in the layer; G^l_{ij} is the Gram Matrix of the ith filter at position j in layer; *OC* is the output of content; *IC* is the input of content; F_l is the number of filters in layer.

3.5 Image Regularization

The reason we add one more step of regularization before stepping to output the combination of texture synthesis and content features is that we want to smooth the pixels in the combination output, and better show the real features captured by our training. Based on this aim, we set total variation as the index for regularization, which bring more piece-wise constant patches to images:

$$R^\beta_{TV} = \sum_{i,j} [(x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2]^{\frac{\beta}{2}} \tag{4}$$

Given the 4 dimension sum up matrix which we mentioned before content loss, the total variation R_{TV}^{β} is shown. Here, *x* is the 4 dimension sum up matrix, *i* is the height value in that matrix, *j* is the width value in that matrix. β is a value can be assigned to any reasonable value. Here, we set β = 1.25,

3.6 Texture Stylization

After we get the loss functions of content and style, we can combine the two images together by following the method proposed by Gatys et al(2015):

$$L_{total} = w_1 \times \sum_{l=1}^{L} L_{content}^{l} + w_2 \times \sum_{l=1}^{L} L_{style}^{l} \tag{5}$$

In this equation, *L* is the total layer number of VGG16 network, while in our practice, we only choose 5 layers, 'block1_conv2', 'block2_conv2', 'block3_conv3', 'block4_conv3','block5_conv3' and used them as the layers for calculating the loss functions. The reason is that these 5 layers lead to the best visualization results. $L_{content}$ is the loss function of content being discussed in the third part; L_{style} is the loss functions of any style showing in the forth part, w_1 = 0.025, w_2 = 1.

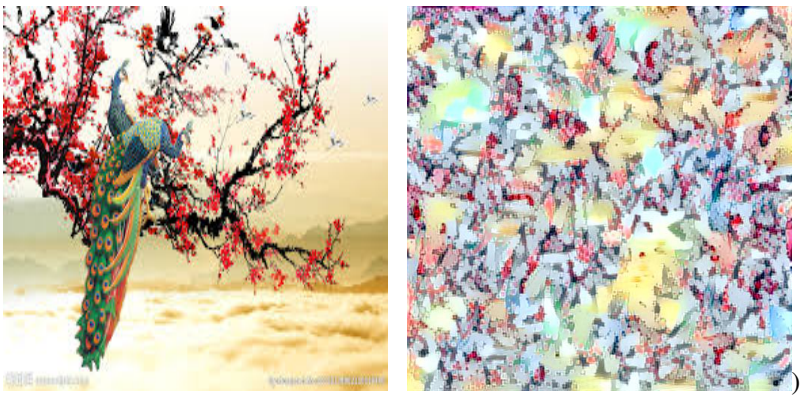
Here, by adding the regularization for processing the output image, we change the above equation to:

$$L_{total} = w_1 \times \sum_{l=1}^{L} L_{content}^{l} + \frac{w_2}{N_L} \times \sum_{l=1}^{L} L_{style}^{l} + w_3 \times \sum_{l=1}^{L} R_{TV}^{\beta} \tag{6}$$

In this equation, *L* is the total layer number of VGG16 network, while in our practice, we only choose 5 layers, 'block1_conv2', 'block2_conv2', 'block3_conv3', 'block4_conv3','block5_conv3' and used them as the layers for calculating the loss functions. The reason is that these 5 layers lead to the best visualization results. $L_{content}$ is the loss function of content being discussed in the third part; $N_L is the number of layers we use here, which is 5; L_{style}$ is the loss functions of any style showing in the forth part, R_{TV}^{β} is the regularization, total variation shown in the fifth part, w_1 = 0.025, w_2 = 0.25, w_3 = 1.

The output graph which combine the features of content and one style is the combination of the loss functions of the content and style.

By letting the w1 to be 0, we output the style reconstruction as images, and the style images are shown as following:



(a) Original Graph (b) Texture Synthesis

Traditional Chinese Painting

(c) Original Graph

(d) Texture Synthesis

Chinese Papercut

(e) Original Graph

(f) Texture Synthesis

Chinese Ink Painting

Figure 1: Texture synthesis results by the method of Gatys et al. (2015)

After comparing the output of the styles, it is not hard to see that the output of Chinese ink painting does not explain the style well. The reason is that there is only two colors, black, and white. The lack of contrast leads to the bad recognition of style in Chinese ink painting. The reason here can also explain why for most previous related works, their sample images shown in their papers are always very colorful.

Then, to follow the work of Gatys, Ulyanov, Luan, Johnson, and so on, we combine our content with only one style, and output 3 combination images as following:

Based on our discussion for the style output, it is easy to explain why the output of the combination of "Venice gondola at night" and "Chinese ink painting" does not fit the style of "Chinese ink painting" well.

One specification for our method is that comparing to works in Gatys, Ulyanov, Luan, Johnson, and so on, we combine two reference styles with a completely different style of input image, thus, our loss function change to:

$$L_{total} = w_1 \times \sum_{l=1}^{L} L^l_{content} + \frac{w_2}{N_L} \times \sum_{l=1}^{L} L^l_{style_1} + \frac{w_3}{N_L} \times \sum_{l=1}^{L} L^l_{style_2} + w_4 \times \sum_{l=1}^{L} R^\beta_{TV} \qquad (7)$$

Figure 2: Content image used in this project (Venice gondola at night)

(a) Traditional Chinese Painting (b) Chinese Papercut (c) Chinese Ink Painting

Figure 3: One style transformation results

In this equation, L is the total layer number of VGG16 network; $L_{content}$ is the loss function of content being discussed in the third part; L_{style_1}, L_{style_2} are the loss functions of any style showing in the forth part, w_1 = 0.025, w_2 = 0.25, w_3 = 0.25, $N_L = 5, whichisthenumberoflayersweselectedtousehere$, w_4 = *1,* ,R_{TV}^{β} is the total variation showing in the fifth part.

4 Result and Conclusion

This project applies multi-style transformation on graphs to generate new artistic works. The image stylization method by Gatys et al. (2015) is improved and complicated to achieve the objective. An example of the multi-style transformation is shown in Figure 4. The content image is as Figure 2. Two art styles from a traditional Chinese painting (Figure 1a) and a Chinese papercut (Figure 1b) were synthesized. The new generated graph in Figure 4 shows a good result of style combination in terms of human recognition.

However, the efficiency of the methodology is relatively low. Each iteration of optimization exerted 10 minutes, and the total iteration to generate Figure 4 was 10. While some researchers have applied

Figure 4: Image stylization with traditional Chinese painting and Chinese papercut

faster optimization algorithms, improving efficiency of image stylization is considered as future work because of time limit.

5 Future Work

One of the disadvantages of using CNN in image stylization is its low efficiency. Because the architecture of pretrained CNN is complicated, extracting styles and contents is relatively slow. Another image generation method is to decrease pixelized losses between output and original images using multi-layer perceptrons (MLP). While this algorithm has better efficiency than using CNN, the sacrifice of accuracy is more dominant. Johnson et al. (2016) combined the benefits of CNN and MLP in image transformation. Images generated by CNN transformation and original content images were considered as output and input data respectively, and training data were from the method of Gatys et al. (2015). Accomplishing image stylization with high efficiency and little sacrifice of accuracy will be the future work of this project.

6 Reference

1. Simonyan, K., & Zisserman, A. (2014). *Very deep convolutional networks for large-scale image recognition.* arXiv preprint arXiv:1409.1556.
2. Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). *A neural algorithm of artistic style.* arXiv preprint arXiv:1508.06576.
3. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *Imagenet classification with deep convolutional neural networks.* In Advances in neural information processing systems (pp. 1097-1105).
4. Van der Maaten, L., & Hinton, G. (2008). *Visualizing data using t-sne.* Journal of Machine Learning Research, 9.
5. Karpathy, A. (2016). http://cs.stanford.edu/people/karpathy/cnnembed/
6. Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., & Lipson, H. (2015). *Understanding neural networks through deep visualization.* arXiv preprint arXiv:1506.06579.
7. Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). *Striving for simplicity:*

The all convolutional net. arXiv preprint arXiv:1412.6806.
8. Zeiler, M. D., & Fergus, R. (2014, September). *Visualizing and understanding convolutional networks.* In European conference on computer vision (pp. 818-833). Springer, Cham.
9. Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). *Deep inside convolutional networks: Visualising image classification models and saliency maps.* arXiv preprint arXiv:1312.6034.
10. Johnson, J., Alahi, A., & Fei-Fei, L. (2016, October). *Perceptual losses for real-time style transfer and super-resolution.* In European Conference on Computer Vision (pp. 694-711). Springer International Publishing.
11. Luan, F., Paris, S., Shechtman, E., & Bala, K. (2017). *Deep Photo Style Transfer.* arXiv preprint arXiv:1703.07511.
12. Efros, A., Leung, T. (1999). *Texture synthesis by non-parametric sampling.* ICCV(pp. 1033-1038).
13. Gatys, L.A., Ecker, A.S., & Bethge, M. (2015). *Texture synthesis using convolutional neural networks.* In: Advances in Neural Information Processing Systems 28.
14. Ustyuzhaninov, I., Brendel, W., Gatys, L. A., & Bethge, M. (2016). *Texture synthesis using shallow convolutional networks with random filters.* arXiv preprint arXiv:1606.00021.
15. Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., & Yang, M.-H. (2017). *Diversified texture synthesis with feed-forward networks.* arXiv preprint arXiv:1703.01664.
16. Jetchev,N., Bergmann,U., & Vollgraf, R.(2016). *Texture synthesis with spatial generative adversarial networks.* arXiv preprint arXiv:1611.08207.