# Spatially-Adaptive Normalization for Human Face Video Synthesis

Yunbai Zhang: yz3386@columbia.edu
Shiyuan Li: sl4398@columbia.edu
Jiao Zhou: jz3071@columbia.edu

## Abstract

*The new conditional normalization method called Spatially-Adaptive Normalization(SPADE) has been proved to be a good fit for landscape and cityscape image synthesis. Since neither of the the three datasets the authors used involved "human faces", we suspect that the SPADE normalization is not suitable to work on "human faces". So our task is to improve the SPADE model to make it applicable to human face dateset. Also, we expand our work by implementing this method on video. The architecture of our work is to do image segmentation first and then feed these segmented images into SPADE trained model. After synthesizing these images as video, we implement the temporal consistency loss function in videos Style-transfer to remove flickers and maintain the consistency of videos.*

## 1. Introduction

Conditional image synthesis is a method that generating images depends on input data. Recently, some researchers use the normalization layer without requiring external data. During this process, the learned affine parameters are used to control the global style of the output and so are uniformly across the spatial coordinates. [1] As a result, this may cause semantic information washing away. Compare to the unconditional normalization method, the parameters for SPADE normalization method have already gained enough information about the label layout, it does not need to feed the segmentation map to the first layer. Therefore, it would maintain more semantic information than transitional normalization layers, and so it would better fit tasks of image synthesis. There are a lot of important applications for extending the SPADE model to human face synthesis, such as face recognition and detecting the misinformation and propaganda. Thus, SPADE for face video is definitely an area which is worth the effort to develop. In addition, our model enables users to choose an artistic style image to control the global appearances of the output video.

### 1.1 Related work

**Conditional Normalization layers** There are some popular unconditional normalization methods, such as Batch Normalization and Instance Normalization. For Batch Normalization, we divide the data into different batches and normalize each batch by its mean and standard deviation. [5] Differently from Batch Normalization, the linear transformation part for SPADE normalization, which is shown on Figure 1, the parameters $\gamma$ and $\beta$ depend on the input segmentation mask and vary with respect to the location (y, x).

**Conditional Normalization layers** Intuitively, different from unconditional normalization layers, the conditional one depends on external data. Unlike prior conditional normalization methods, $\gamma$ and $\beta$ in SPADE Normalization method are not vectors, but tensors with spatial dimensions. The produced $\gamma$ and $\beta$ are multiplied and added to the normalized activation element-wise. [1]

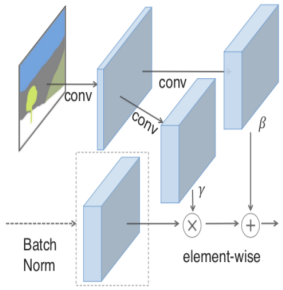



Figure 1: The Structure for SPADE Normalization

### 1.2 Problem formulation and goal

The goal of our project is to use generative adversarial networks model(GAN) to synthesize human face videos. Suppose the condition that if we only have one segmentation mask with a single label as input, the convolution outputs will be uniform with different labels having different uniform values.[1] The normalized activation will become all zeros, and the feature of human faces will largely lose. Thus, We plan to introduce SPADE method which can maintain semantic face feature information. In addition, we have learned that the traditional omission of noise leads to featureless "painterly" look. We tried to avoid it on our generator.

## 2. Method

**2.1 Method Description** Suppose that $\mathbf{m} \in L^{H \times W}$ to be a semantic mask, where $L$ denotes semantic labels, and $H$ and $W$ are the image height and width. Our task is to convert an input segmentation mask m to a realistic image. [1] Let $H^i$ and $W^i$ denote the height and width of the activation map. The method of computation for SPADE is similar to that of Batch Normalization, in which the activation is normalized in channel pairwisely. And the activation value

$$\gamma^i_{c,y,x}(m)\frac{h^i_{n,c,y,x} - \mu^i_c}{\sigma^i_c} + \beta^i_{c,y,x}(m)$$

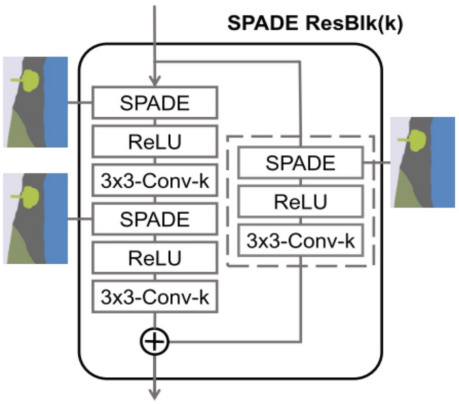
where the mean and variance in the channel c is given by

$$\mu^i_c = \frac{1}{NH^iW^i}\sum_{n,y,x} h^i_{n,c,y,x}$$

and [1]

$$\sigma^i_c = \sqrt{\frac{1}{NH^iW^i}\sum_{n,y,x}\left(h^i_{n,c,y,x}\right)^2 - \left(\mu^i_c\right)^2}$$

We use the SPADE normalization layer to construct the Resnet block, which we will use in the generator. We construct a stack of three layers involving SPADE normalization, RELU activation fuction, and one convolutional layer. The structure of the Resnet block is shown as follows:




**2.2 Data Description** We used a new dataset of human faces called Flickr-Faces-HQ (FFHQ), consisting of 70,000 high-quality images at 10242 resolution. The dataset includes vastly more variation than CELEBA-HQ in terms of age, ethnicity and image background, and also has much better coverage of accessories such as eyeglasses, sunglasses, hats, etc [2]. We split the whole dataset into two parts, 20% as test data, 80 % as training data. Figure 2 shows samples from the dataset.

**2.3 Data Pre-processing** In order to generate more accurate segmentation map, we first slice the input video into 152 images with the frame rate 15. After that, we crop and re-size each original frame into size 512*512 such that these

Figure 2: Samples from FFHQ dataset

would match the segmentation outputs. Then we remove the random noise on segmentation images. For example, at some pixel, it should be labeled in "background" but the algorithm misclassified it in label "hat" "cloth" or "skin". Therefore, in that case, we fix this kind of misclassification by checking the region and the labels. Finally, according to these segmentation labels, we uniform the background into white, thereby generating better synthesis results.

**2.4 Image Segmentation** In order to implement SPADE method, we need to use the segmented pictures to do image synthesis. So, we firstly need to transform our original images to segmented ones. We use $2D$ convolutional network to translate the original image to a segmentation map. For each block, we firstly use Batch Normalization followed by $Relu$ function and a convolutional layer. We have total 3 blocks. There are 19 classes for our segmentation model, and the classes table is shown on Figure 6 (page 3). The accuracy for this model is 93.41 percent. [3]. For details, after training this model, we label each part of face as Figure 6 shows. And then we synthesis each part to a segmentation image with different colors. Figure 3 shows one example of this transformation.

Figure 3: Original image (left) and Segmentation image (right)

### 2.5 Training Process

**2.5.1 Generator** After the linear mapping, we did some reshaping such that it could fit the SPADE ResBlock. For each layer, we use a SPADE normalization followed by a ReLU activation and 3*3 convolutional layer with k filters. We repeat this process several times to finish SPADE ResBlk construction. The architecture of the generator is shown on Figure 4

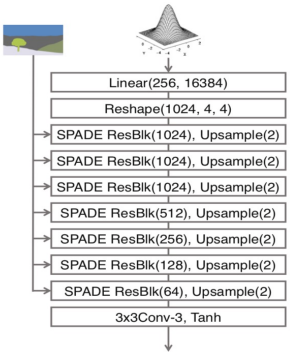



Figure 4: Architecture of Generator

**2.5.2 Discriminator** For the discriminator, we apply the method that is used in the pix2pixHD paper, which is a multi-scale design. We first downsample the real and synthesized high-resolution images by a factor of 2 and 4 to create an image pyramid of 3 scales.[4] Then it takes the segmentation map and the synthesized image from the generator as input. In each layer, it follows by a 4*4 convolutional layer, an instance normalization and LReLU activation function. For the last layer, corresponding to Patch-GAN, we only use convolutional layer. The architecture of discriminator is shown on figure 5.

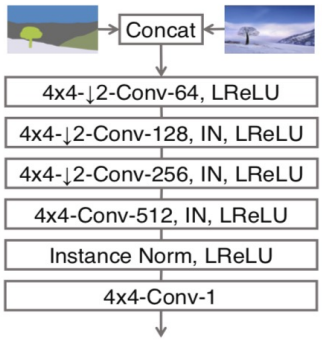



Figure 5: Architecture of Discriminator

**2.5.3 Encoder** The image encoder consists of several stride-2 convolutional layers with 3*3 filters. The last layer is two linear one, in which we can get the mean vector $\mu$ and the variance vector of $\sigma^2$

**2.5.4 Learning Objective** When training the proposed framework with the image encoder, we include a KL Divergence loss: $\mathcal{L}_{\text{KLD}} = \mathcal{D}_{\text{KL}}(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$, where the prior distribution $p(z)$, and z is from standard normal distribution and q is determined by a mean vector and a variance vector. For this case, we weight for the KL-Divergence loss is 0.05

**2.5.5 Training process setting** In the process of segmentation, we use the pre-trained model provided by

| Label list | | |
|---|---|---|
| 0: 'background' | 1: 'skin' | 2: 'nose' |
| 3: 'eye_g' | 4: 'l_eye' | 5: 'r_eye' |
| 6: 'l_brow' | 7: 'r_brow' | 8: 'l_ear' |
| 9: 'r_ear' | 10: 'mouth' | 11: 'u_lip' |
| 12: 'l_lip' | 13: 'hair' | 14: 'hat' |
| 15: 'ear_r' | 16: 'neck_l' | 17: 'neck' |
| 18: 'cloth' | | |

Figure 6: Label List of segmentation map

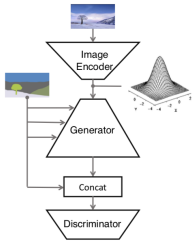



Figure 7: Architecture for the training model [1]

CelebAMask-HQ. Then we feed the segmented images as input, use SPADE normalization and generate the synthesized output. During the training process, we use 20000 segmented images as input and set our training epoch to be 30 with 1000 iterations per epoch. During our training process, we use NVIDIA DGX1 with 8 V100 GPUs.

Unlike the state of art generator method pix2pixHD which includes both down-sampling layers and up-sampling layers, our SPADE generator can achieve a better result by removing the down-sampling layers. We start by using a random vector as our input. Then we use an encoder to process a real image into this vector, which then is fed to our SPADE generator. The encoder and generator form a variational autoencoder. This encoder has the ability to capture the style of the image. Then the generator will use the encoded style and the segmentation mask through the SPADE to create an image. Finally, we plan to add some random noises and novel mixing regularization to make neighboring styles less correlated and our image more realistic. When we train the model, we simply use two layers of convolutional neural network for each residual block shows how we use the SPADE method in training our model.

**2.5.6 Performance Criteria.** We use mean Intersection-over-Union(mIoU) and segmentation accuracy to measure the performance of our model.[1] Here is the formula for mIoU criteria:

$$IoU = \frac{target \cap predictor}{target \cup predictor}$$

where higher mIoU score corresponds to higher similarity between the ground truth and the predictor. The formula for segmentation accuracy is:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

A true positive (TP) represents a pixel that is correctly predicted to belong to the given class (according to the target mask) whereas a true negative (TN) represents a pixel that is correctly identified as not belonging to the given class. A false positive(FP) represents wrongly identified the pixel that belongs to the given class; and a false negative represents wrongly identify the class that does not belong to given class.

**2.6 Testing Process for SPADE model.** For the testing part, we first segment the original image, feed the segmented image to the training model, and then generate a synthesized image. The second step is that we use the new synthesized image as input and implement the segmentation algorithm to generate a new segmented image. Our test algorithm is that we use the mIoU and segment accuracy as performance criteria and test whether these two segmented image are similar enough. The reason we use this algorithm is that it is hard to compare accuracy between the original one and the synthesis one, but if we use segmentation images, it is easy for us to detect the attributes and location of facial features by implementing features classification.
The architecture of our test algorithm is shown on Figure 8.

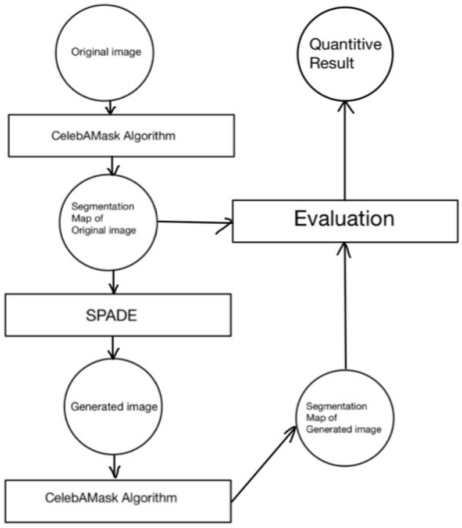



Figure 8: Testing algorithm

**2.7 Synthesize video by SPADE normalization**

**2.7.1 The Goal for Synthesize video** By using the model that showed above, we can synthesize fake human faces corresponding to the input segmentation images. What about generating synthesized face video? Our work expand from generating synthesized images to videos.

**2.7.2 The difficulty for Video Synthesis** If we directly combine those synthesized images into video, the changes from frame to frame would be dramatic, thereby resulting in a disturbing flickering effect. The reason is that most of the frames are the same, and only a few of them have new information. Thus, if we treat each image independently and ignore the inter-communication between frames, we would get flickering video.

**2.7.3 The method for maintaining temporal coherence** To enforce stronger consistency between adjacent frames, besides style loss and content loss, we introduce a new consistency penalty, that is the temporal consistency loss. Basically, the effect of the temporal coherence loss is that if we color this person this way a moment ago, it cannot change the style and content of the region drastically. Theoretically, the temporal consistency loss function penalizes deviations from the warped image in regions where the optical flow is consistent and estimated with high confidence. [8] The formula for temporal consistency loss is as following

$$\mathcal{L}_{temporal}(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{c}) = \frac{1}{D}\sum_{k=1}^{D} c_k \cdot (x_k - \omega_k)^2$$

where $\boldsymbol{c} \in [0,1]^D$ is the per-pixel weighting of the loss and $D = W \times H \times C$ is the dimension of the image. In our case, the dimension is D = 256* 256* 152. We use the weights $0$ and $1$ to incorporate the certainty of the optical flow prediction. And the short term overall loss function becomes

$$\begin{aligned}\mathcal{L}_{\text{shortterm}}\left(\boldsymbol{p}^{(i)}, \boldsymbol{a}, \boldsymbol{x}^{(i)}\right) &= \alpha\mathcal{L}_{\text{content}}\left(\boldsymbol{p}^{(i)}, \boldsymbol{x}^{(i)}\right) \\ &+ \beta\mathcal{L}_{\text{style}}\left(\boldsymbol{a}, \boldsymbol{x}^{(i)}\right) \\ &+ \gamma\mathcal{L}_{\text{temporal}}\left(\boldsymbol{x}^{(i)}, \boldsymbol{\omega}_{i-1}^{i}\left(\boldsymbol{x}^{(i-1)}\right)\right)\end{aligned}$$

where $x_{i-1}$ indicates the last frame that we stylized. [8]

Since for the video we not only want to penalize deviations from the previous frame, but also from temporally more distant frames, so we want to introduce the long-term temporal consistency loss function. The formula is given by:

$$\begin{aligned}\mathcal{L}_{\text{longterm}}\left(\boldsymbol{p}^{(i)}, \boldsymbol{a}, \boldsymbol{x}^{(i)}\right) &= \alpha\mathcal{L}_{\text{content}}\left(\boldsymbol{p}^{(i)}, \boldsymbol{x}^{(i)}\right) \\ &+ \beta\mathcal{L}_{\text{style}}\left(\boldsymbol{a}, \boldsymbol{x}^{(i)}\right) \\ &+ \gamma \sum_{j \in J: i-j \geq 1} \mathcal{L}_{\text{temporal}}(\boldsymbol{x}^{(i)}, \omega_{i-j}^{i})\end{aligned}$$

It is similar to the short term one except there is one more parameter $j$ that indicates the indices that each frame should take into account, and by summing over all $j$s we get the long term consistency loss.

**2.7.4 Segmentation map cleaning** Since adding the temporal loss function can only reduce the trivial inconsistency, but for the video that generated from SPADE normalization model, the background color was messed. In order to improve the video, before feeding to the temporal coherence model, we do the image cleaning again. We put the generated images into the segmentation model again. Then according to the labels of the segmentation maps, we uniform the background color so that the background color would not change dramatically.

**2.7.5 Style transfer to human face** Besides temporal loss function, We have two inputs for neural style transfer. One is the style which we wish to apply to the generated image;

the second one is content. We calculate the dissimilarity between the generated images and the style images. To do so, we used a matrix called the Gram matrices. A Gram matrix results from multiplying a matrix with the transpose of itself.

$$G^l_{i,k} = \sum_k F^l_{i,k} F^l_{j,k}$$

where $F$ is the vectorized feature map. In one given layer, the Gram matrix is the inner product of two feature maps. This Gram matrix can reflect information such as texture, shapes, and weights. Then, after we calculate the gram matrix, we calculate the euclidean distance between those two matrices to see how similar two images' style are. After that, we used the Adam optimizer to minimize our loss. The following is the formula to calculate this style loss.

$$\mathcal{L}_{style} = \sum_l \sum_{i,j} \left(\beta G^{s,l}_{i,j} - \beta G^{p,l}_{i,j}\right)^2$$

**2.7.6 How Style transfer implement on our video** In order to evaluate our short-term temporal loss, we combine videos of three of our group members and five artistic paintings as our style input, which are Scream, Wave, Rain princess, Udnie and Wreck. We set the frame rate at 20 frames per second. For the test part, We compare the stylized image of one frame with the stylized image of its previous frame to find out the regions that are not inconsistent, then we calculate their mean square error.

After we apply the temporal coherence into our generated images, we get the following result for applying temporal coherence with different styles.

| Scream | Wave | Rain princess | Udnie | Wreck |
|---|---|---|---|---|
| 0.0065 | 0.0097 | 0.0032 | 0.0054 | 0.0021 |

Table 1: MSE of Different Style

If we clean the background of the images before applying the temporal coherence, we can see a significant improvement of MSE of all styles.

| Scream | Wave | Rain princess | Udnie | Wreck |
|---|---|---|---|---|
| 0.0043 | 0.0086 | 0.0026 | 0.0020 | 0.0009 |

Table 2: MSE of Different Style after we process the images

**2.8 Summarize for our method for video synthesizing** Given the input video, after doing data pre-process, we feed each frame to segmentation model. Then through the labels that generated by segmentation model, we clean the original image again and feed into SPADE model. After that, by adding the temporal coherence loss function and cleaning the synthesized image, we can finally generate a stable synthesized video.
The architecture of the video synthesis is in Figure 9.

## 3.Result

**3.1 Result for image** From the image above, we can see that the segmented image generated by the original one is very closed to that of generated by the synthesized one.

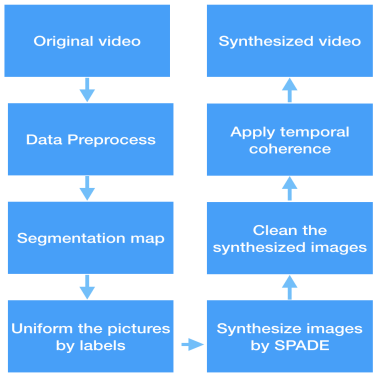



Figure 9: Overall Video Synthesize Procedure

Figure 10: Original image (Left), segmentation image (middle) and segmentation image generated by new synthesized image(right)

| | Accuracy | mIoU |
|---|---|---|
| pix2pix with SPADE | 0.967 | 0.85 |
| pix2pixHD++ | 0.94 | 0.79 |
| pix2pix | 0.902 | 0.71 |

Table 3: Test for Human Faces

**Analysis Result:** After running our evaluation algorithm on SPADE, we got an accuracy score of 0.967 and mIoU of 0.85. As we can see from the results, the model with SPADE out-performed the pix2pix model in terms of both accuracy and mIoU. In order to ensure that our superior performance is due to SPADE normalization, we also run a test on a pix2pixHD++ model. This model has all the same attribute except that it uses the batch normalization. For example, like our SPADE, it also does not contain down-sampling layers.

In addition, even though SPADE outperform both in pix2pix and pix2pixHD++ model when applying to human face data-set, three models' performances are considerably better than when they are applied to COCO-Stuff dataset, ADE20K, and Cityscapes. Those results are cited from a study done in UC Berkeley when they applied three models on the landscape.[1]

| | Accuracy | mIoU |
|---|---|---|
| pix2pix with SPADE | 0.679 | 0.385 |
| pix2pixHD++ | 0.458 | 0.146 |

Table 4: Test for Landscape

It is actually not surprising that SPADE model works bet-

ter on human faces, even though all of these methods work better on human face dataset than on street senses and landscapes dataset. The reason is that for the human face dataset, we only use 19 classes in the segmentation map. However, for street senses and landscapes, it has more than 200 classes. As a result, when performing on treet senses and land-scapes data, it is more likely to make mistakes. The results prove our hypothesis that SPADE would be an effective model for human face synthesis.

Figure 11: Screen Shot of the original synthesized video without temporal consistency and style (the first row)
stylized synthesized videos with temporal consistency (the second, third, fourth and fifth rows)

### 3.2 Result for video

**Analysis for video result:** By processing segmentation map cleaning, we have uniformed the background and reduced the misclassified labels; and by adding the temporal consistency loss function, we allow some waves of light changes. Therefore, after these steps, our synthesized videos become much more consistent and stable.

## 4. Limitation of our work and future improvement

**Change of the shape of the face** Since the model that we used to synthesize images is Pix2Pix, that is for static image, but after that we expand our work to videos. In a video for human face, the correlation among frames is very high, but by our training model we just treat the synthesized image independently. As a result, the shape of the face would change a lot. Therefore, for future work, we can add the Long short-term memory (LSTM) or Recurrent neural network (RNN) to maintain the connection among frames.

**Our label class** Since there are only 19 labels defined our segmentation model, if we have some labels other than these, this algorithm cannot detect them but assign them to another classes. For example, if someone wears earring or glasses, this algorithm cannot identify them, and it would mistakenly classify the earring into part of ear and eyes. As a result, the person in our generated one would not have earring. Thus, for our future work, we can enrich the classes in our segmentation algorithm so that it can do recognition more precisely.

This is our github link: https://github.com/summerdeeplearning/deep-learning

## 5. References


[1] Taesung Park, Ming-Yu Liu, Ting-Chun, Wangun,Yan Zhu; Semantic Image Synthesis with Spatially-Adaptive Normalization. *IEEE: Computer Vision and Pattern Recognition*. 2019

[2] Karras, T., Laine, S. Aila, T. A Style-Based Generator Architecture for Generative Adversarial Networks. *Neural and Evolutionary Computing (cs.NE)*. 2018

[3] Ziwei Liu, Ping Luo, Xiaogang Wang and Xiaoou Tang; "Deep Learning Face Attributes in the Wild" *IEEE International Conference on Computer Vision (ICCV)*. 2015

[4] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, Bryan Catanzaro; "High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs" *Computer Vision and Pattern Recognition*. 2018

[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.

[6] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance nor- malization: The missing ingredient for fast stylization. *arXiv preprint*. 2016

[7] V. Dumoulin, J. Shlens, and M. Kudlur. A learned repre- sentation for artistic style. *International Conference on Learning Representations (ICLR)*. 2016.

[8] M.Ruder, A.Dosovitskiy, T. Brox Artistic style transfer for videos. *Computer Vision and Pattern Recognition 2016*