Generating Cartoon Style Facial Expressions with StackGAN

Xiaoyi Li
Stanford University
xiaoyili@stanford.edu

Xiaowen Yu
Stanford University
wyu1207@stanford.edu

Abstract

Facial expression generation and photo-to-cartoon transfer are two interesting and useful tasks. A series of approaches have been proposed to tackle them separately. This project proposes an end-to-end stacked jointly learning architecture for these two tasks. With two methods to stack the two GANs, we explore whether the order of GAN will have influence on the quality of final outputs. Our experiments show that transferring the original face photo to cartoon style before generating multiple expressions will provide outputs with slightly better quality.

1. Introduction

Facial expression generation transforms the expression of a given face photo to a target one without affecting the identity properties and has applications in facial animation, human-computer interactions, entertainment, etc. Photo-to-cartoon transfer is a task that transfers style of real-world photo to cartoon and its applications range from artistic interests, publication in media to children education. These two tasks are naturally connected since it is interesting to see the cartoonization of face photos with different expressions.

In this project, we propose an end-to-end stacked jointly learning architecture for these two tasks. A series of statistical learning approaches have been proposed to tackle the two tasks separately. For example, StarGAN model (Choi et al., 2018) transforms the face image to multiple new expressions based on the high-level semantic understanding of the input image. CartoonGAN (Chen et al., 2018) proposes a solution to transforming photos of real-world scenes into cartoon style image. However. we think they share one fundamental characteristic essentially and the performance will be improved if we train the two GANs jointly. Therefore, we stack StartGAN and CartoonGAN to complete the two tasks together. One more interesting question is that whether the order of each GAN in the stacked model has impact on the quality of final output. In other words, whether we should transfer the style first or generate the multiple expressions first. We propose two ways to stack the two GANs sequentially and aim to test which provides better results. To the best of our knowledge, this is the first research that tries to explore this topic.

2. Dataset

2.1. RAF-DB

The database we used to generate facial expressions is the Real-world Affected Faces Database (RAF-DB). The RAF-DB contains 12271 training samples and 3068 testing samples from real-world images with 7-dimensional expression categories (Surprise/ Fear/ Disgust/ Happiness/Sadness/Anger/Neutral). In this project we use the aligned images which are pre-processed with size of 100x100 pixels.

2.2. IIT-CFW

We use IIT-CFW dataset to train our own face photo to cartoon style transfer model. This dataset is a large set of cartoon faces in the wild harvested from Google image search and contains 8,928 annotated cartoon faces of famous personalities of the world with varying professions.

3. Approach

3.1. Separate models

We trained StarGAN for facial expression generation and CartoonGAN for cartoon style transfer separately. We modified the original StarGAN model and trained it with RAF-DB data. The architecture of the model is shown in Fig.1. As illustrated below, we then input the sample photos into the trained model and get the GAN-generated-expressions photos.

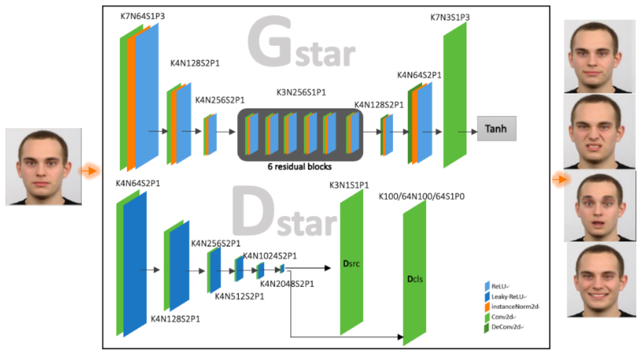


Fig.1 Architecture of StarGAN

The loss function of the StarGAN model has two parts:

$$
\begin{aligned}
L_{StarGAN-D} &= -\mathbb{E}[D_{src}(x_i)] + \mathbb{E}[D_{src}(G(x_i,c))] \\
&\quad + \lambda_{gp}\mathbb{E}[(\|\Delta D_{src}(\hat{x}_i)\|_2 - 1)^2] + \lambda_{cls}\mathbb{E}[-logD_{cls}(c'|x_i)] \\
L_{StarGAN-G} &= \mathbb{E}[D_{src}(x_i)] - \mathbb{E}[D_{src}(G(x_i,c))] \\
&\quad - \lambda_{gp}\mathbb{E}[(\|\Delta D_{src}(\hat{x}_i)\|_2 - 1)^2] + \lambda_{cls}\mathbb{E}[-logD_{cls}(c'|G(x_i,c))] \\
&\quad + \lambda_{rec}\mathbb{E}[\|x - G(G(x_i,c),c')\|_1]
\end{aligned}
$$

We trained CartoonGAN model with IIT-CFW dataset for photo-to-cartoon style transfer. The architecture of the model is shown in Fig.2.

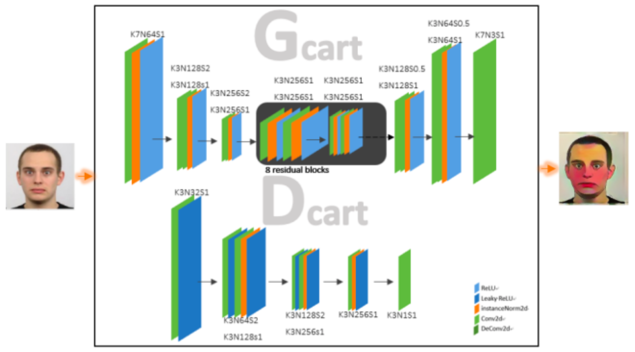


Fig.2. Architecture of CartoonGAN

The loss function of the CartoonGAN is as follows:

$$
\begin{aligned}
L_{CartoonGAN} &= L_{adv}(G_2,D_2) + \omega L_{con}(G_2,D_2) \\
&= -\mathbb{E}[logD(G_2(x_i))] - \mathbb{E}[\log(1 - D(G_2(p_k))] \\
&\quad - \omega\,\mathbb{E}[\| VGG_l(G_2(p_i)) - VGG_l(p_i) \|_1]
\end{aligned}
$$

For the first part Adversarial loss, the goal of training the discriminator D is to maximize the probability of assigning the correct label to generated image such that generator G can be guided correctly by transforming the input to the correct manifold. The second part Content loss is to ensure the resulting cartoon images retain semantic content of the input photos. We adopt the high-level feature maps in the VGG pre-trained model.

3.2. Stacked models

3.2.1 Architecture

The two GANs are stacked together and we train the stacked model end-to-end. We propose two methods to stack GANs: (1) StackGAN A: StarGAN is considered as the first GAN and CartoonGAN as the second, and (2) StackGAN B: the order of GANs is flipped. Here both methods use RAF-DB as StarGAN training dataset and IIT-CFW for CartoonGAN. More specifically, in StackGAN A, the original face photo is trained to generate expressions and then trained to transfer to cartoon style in each iteration. While for StackGAN B, the original photo is trained to transfer style and then trained to generate multiple expression in each iteration,. In addition to the new architecture, this project aims to explore which way to stack GANs will output better results.

The architecture of our stacked model is shown in Fig.3.

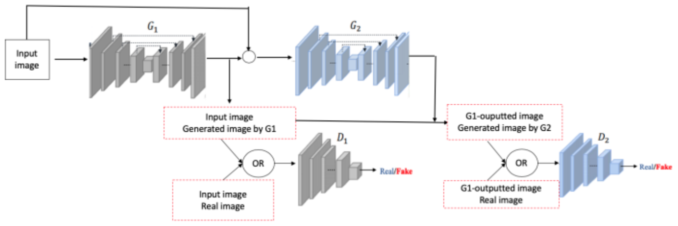


Fig.3. Architecture of StackGAN

3.2.2 Loss function

For StackGAN A with order StarGAN-CartoonGAN, the loss function of the first GAN is the same with StarGAN in separate model. While the generator of the second GAN takes the generated images from the first GAN as inputs, so the loss function of the second GAN is changed accordingly.

$$
L_{GAN_1}(G_1,D_1) = L_{StarGAN}(G,D)
$$

$$
\begin{aligned}
L_{GAN_2}(G_2,D_2|G_1) &= -\mathbb{E}[logD(G_2(G_1(x_i)))] - \mathbb{E}[\log(1 - D(G_2(p_k))] \\
&\quad -\omega\,\mathbb{E}[\| VGG_l(G_2(p_i)) - VGG_l(p_i) \|_1]
\end{aligned}
$$

For StackGAN B with order CartoonGAN-StarGAN, the loss function of the first GAN is the same with CartoonGAN in separate model. Similar with StackGAN A, The generator of the second GAN will take the generated images from the first GAN as inputs.

$$
L_{GAN_1}(G_1,D_1) = L_{CartoonGAN}(G,D)
$$

$$\begin{aligned} L_{GAN_2}(G_2|G_1) = {} & \mathbb{E}[D_{2\_src}(x_i)] - \mathbb{E}[D_{2src}(G_2(G_1(x_i),c))] \\ & - \lambda_{gp}\mathbb{E}\left[\left(\|\Delta D_{2\_src}(\hat{x}_i)\|_2 - 1\right)^2\right] + \lambda_{cls}\mathbb{E}[-log D_{2\_cls}(c'|G_2(G_1(x_i),c))] \\ & + \lambda_{rec}\mathbb{E}[\|x - G_2(G_2(G_1(x_i),c),c')\|_1] \end{aligned}$$

$$L_{GAN_2}(D_2|G_2) = L_{StarGAN-D}$$

The final loss function of the stacked model in both Stacked Model (1) and (2) is the same and as follows.

$$L = \lambda_1 L_{GAN_1}(G_1, D_1) + \lambda_2 L_{GAN_2}(G_2, D_2|G_1)$$

3.3. Hyper parameters

Table 1 shows the hyper parameters we used for training StackGAN.

Table 1. Hyperparameters for StackGAN

Hyperparameters	
image size	68
number of convolutional filters in the first layer of Gstar	64
number of convolutional filters in the first layer of Dstar	64
number of residual blocks in Gstar	6
number of strided convolutional layers in Dstar	6
weight for domain classification loss for StarGAN	1
weight for reconstruction loss for StarGAN	10
weight for gradient panelty for StarGAN	10
mini-batch size for StarGAN	16
number of total iterations for training Dstar	100,000
number of iterations for decaying learning rate for StarGAN	50,000
learning rate for Gstar	0.0001
learning rate for Dstar	0.0001
number of Dstar updates per each Gstar update	5
beta1 for Adam optimizer for StarGAN	0.5
beta2 for Adam optimizer for StarGAN	0.999
learning rate update step for StarGAN	1000
mini-batch size for CartoonGAN	4
epochs for CartoonGAN	100
content lambda for CartoonGAN	0.4
style lambda for CartoonGAN	25
adversarial lambda for Gcart	8
adversarial lambda for Dcart	1
sample size for CartoonGAN	8
learning rate for Gcart	0.00001
learning rate for Dcart	0.00001
learning rate for pretrained VGG	0.00001
pretrain epochs for CartoonGAN	1
lamda1 for first GAN in total loss function	1
lamda1 for second GAN in total loss function	1

4. Results

We implemented our StackGANs using both Torch and Tensorflow 2.0 and in Python language. All experiments were performed on an NVIDIA 16G GPU. We test StackGAN A and StackGAN B with the same set of original face photos in the RAF-DB test set with neutral expression (in total 680 photos). Both models will generate new face photos with multiple expressions in cartoon style. We will present selected results for both models and compare them in Evaluation part.

4.1. Training time

For each epoch the training time for StackGAN A and StackGAN B is similar, both are about 3300s. Here one epoch is 1000 iterations. Since StarGAN needs much more iterations than CartoonGAN, in StackGAN A, StarGAN generates intermediate outputs for CartoonGAN every 1000th iteration, while in StackGAN B, CartoonGAN generates intermediate outputs for StarGAN every iteration. Saving intermediate outputs takes some time so the total training time for StackGAN B is longer than StackGAN A.

4.2. Convergence

For both StackGAN A and StackGAN B, we run 100,000 iterations. The convergence plots from tensorboard are shown as follows. Fig.4 displays the StackGAN A convergence plot for Gstar and Dstar. Fig.5 shows the StackGAN B convergence plot for Gstar and Dstar. As we observe, both models become stable after 100,000 iterations. There is no big difference in the convergence speed.

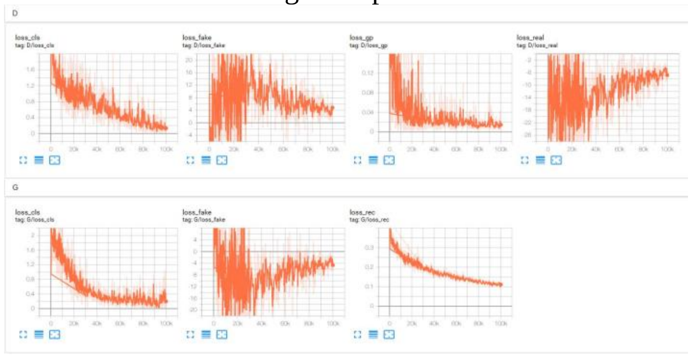


Fig.4. Convergence plot for StackGAN A

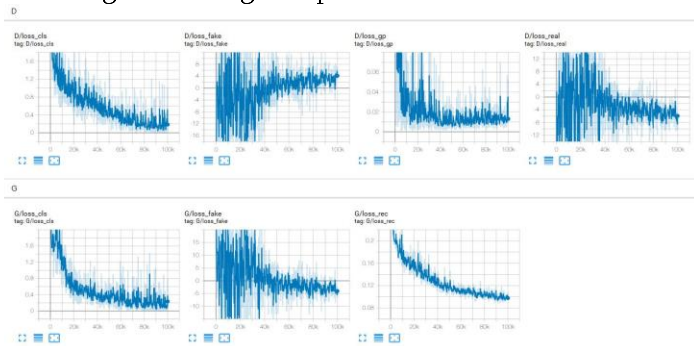
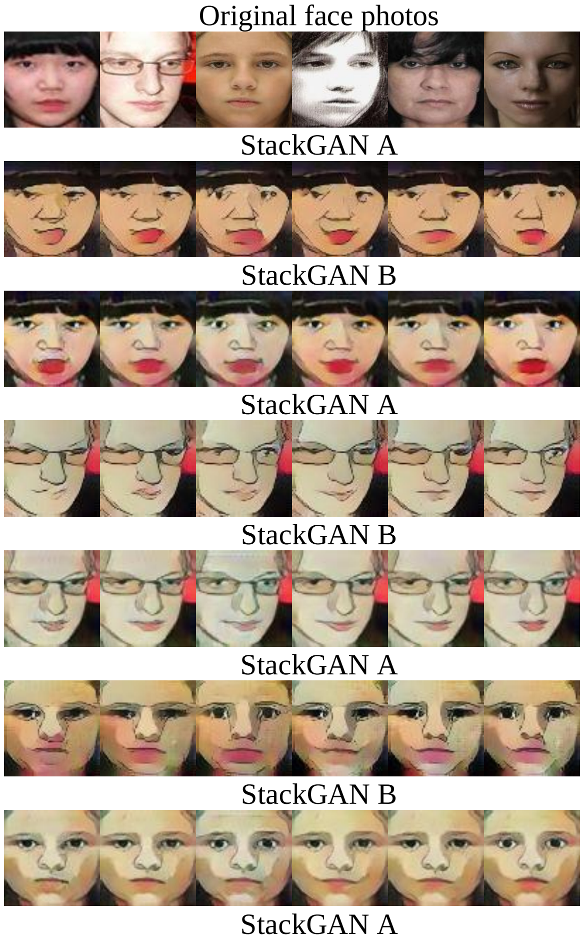


Fig.5. Convergence plot for StackGAN B

4.3. Evaluation

4.3.1 Qualitative evaluation

The Fig.6 displays both the original photos and cartoon expression from StackGAN A and B. The expressions from left to right are anger, disgust, fear, happiness, sadness and surprise. As seen in Fig. 6, StackGAN A and B both properly maintain the personal identity and facial features. But it seems that StackGAN B generates more natural-looking expressions and cartoon styles. We believe that the superiority of StackGAN B in the image quality is due to that style transfer has implicit feature augmentation effect. CartoonGAN produce high-quality cartoon style photos by reproducing the necessary clearly edges and smooth shading while retaining the content of the original photo. Therefore, in StackGAN B, StarGAN can take advantage of the feature augmentation effect by CartoonGAN and generates better expression photos compared with in StackGAN A. However, we do notice that our StackGANs are unable to generate very differentiating disgust, fear and surprise expressions. This is mostly caused by the fact that these expressions in the training dataset itself is not differentiating.



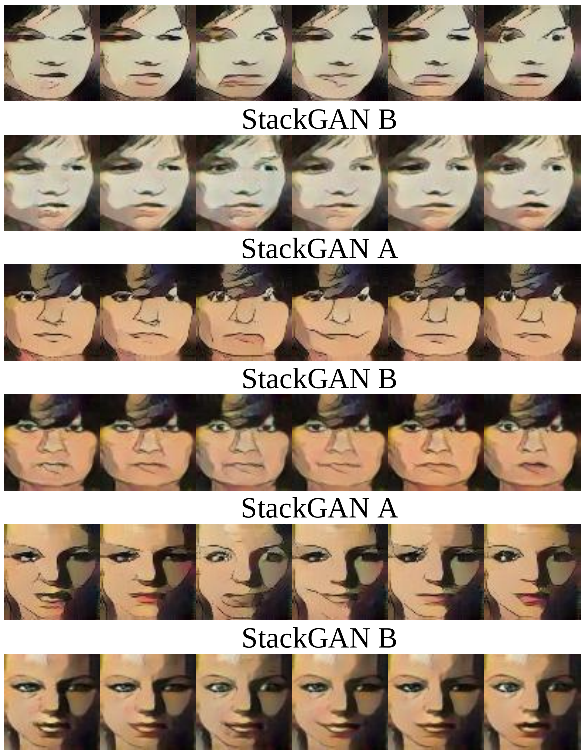


Fig.6. Sample results by StackGAN A and B

4.3.2 Quantitative evaluation

For a quantitative evaluation, we conducted a survey in which we ask people to classify the expression of our output photos manually and then score on the expression quality and cartoon quality. To ensure we get the subjective opinion, we designed two versions of questionnaire A and B with each contains the 15 outputs from either StackGAN A or StackGAN B generated for the same original face photos. Moreover, we included only the anger, happiness and sadness for the same reason we mentioned in 4.2.1. The source of outputs is masked and equally distributed in each questionnaire. The links to questionnaires are in Appendix part. We collected 40 response in total with 22 from questionnaire A and 18 from questionnaire B.

Table 2 shows the average scores on expression quality and cartoon quality. We can see that StackGAN B has higher scores on both expression quality and cartoon quality and this is consistent with our own observations. Table 3 shows the accuracy of the expressions in our output photos. It shows that except for sadness, StackGAN B has higher accuracy on other expressions. On average, StackGAN B's expressions classification accuracy is 6% higher than StackGAN A's. Again, this is consistent with quality scores and our own observations.

Table 2. Average Scores on Quality

Quality Scale (1-5)	StackGAN A	StackGAN B
Expression Quality	3.15	3.32
Cartoon Quality	3.06	3.25

Table 3. Accuracy of expressions

Accuracy	anger	happiness	sadness	Total
StackGAN A	72%	76%	89%	79%
StackGAN B	86%	84%	86%	85%

5. Discussions and Future Work

From the results and comparison of StackGAN A and StackGAN B, we conclude that

(1) Almost no time difference in terms of training each epoch for both StackGANs. However, since the output frequency for the intermediate photos prepared for next GAN training is higher, the actual training time for StackGAN B is slightly longer than StackGAN A.

(2) No significant difference on the convergence rate for both StackGANs.

(3) Switching the training sequence has some impact on the final output photos. Our survey assessing both quality and accuracy of the outputs shows that people tends to prefer StackGAN B a little more. We believe this superiority is caused by the fact that style transfer has implicit feature augmentation effect.

During the training and analyzing results, we do notice a few points that we would like to work on given more time.

(1) The quality of training data is the key to the generated output quality. Unfortunately, we notice the image quality from RAF-DB is not very clear since the dataset is prepared for classification task not for GAN task. If we can get clearer images, we are able to generate better result.

(2) Running either StackGAN A or StackGAN B requires at least a 16GB GPU. We believe some steps in the architecture is not necessary. If given more time, we would like to optimize the architecture to make the model able to run in a less memory-consuming setting.

(3) The architecture of StarGAN and CartoonGAN share similar layer structure. We believe if we add some more layers, such as Maxpool layer to preserve some high-level features in the expression before we send it to style transform, we can get better outputs.

Appendix

[1] Questionnaire links:
https://forms.gle/SkHSP2fkmTouQ8Co9 and
https://forms.gle/YQnm6gCQMJttPUih8.
[2] Github page:
https://github.com/XiaoyiLi-sf/CS230_StackedGAN
[3] Video page:
https://www.youtube.com/watch?v=Z0xXroyExUQ

References

[1] Li, S and Deng, Wand Du, J. 20117. Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[2] Choi Y., Choi M., Kim M., Ha JW., Kim S., Choo J. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. arXiv preprint arXiv:1711.09020v3

[3] Chen Y., Lai YK., Liu YJ. 2018. CartoonGAN: Generative Adversarial Networks for Photo Cartoonization. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR).