Animated Face Image Generation using GANs

Iqra Bismi
Department of Applied Data Science
San Jose State University
San Jose, USA
iqra.bismi@sjsu.edu

Priya Khandelwal
Department of Applied Data Science
San Jose State University
San Jose, USA
priya.khandelwal@sjsu.edu

Saniya Vijay Lande
Department of Applied Data Science
San Jose State University
San Jose, USA
saniya.lande@sjsu.edu

Shilpa Shivarudraiah
Department of Applied Data Science
San Jose State University
San Jose, USA
shilpa.shivarudraiah@sjsu.edu

***Abstract*—In this paper, we propose ArtifyGAN to transform photos of real images into cartoon-style images, which is valuable and challenging in computer vision and computer graphics. The primary goal of this paper is to leverage GAN methodologies to create a dynamic and expressive animated representation of a given facial image. However, existing face animation technologies in the entertainment and gaming industries are inefficient, requiring multiple reference images and unable to preserve the structure of the image. As a result, animations may appear unrealistic. The reason why the existing approach does not yield satisfactory results for cartoonization because cartoon styles have unique features with high-level simplification and abstraction and, cartoon images typically have clear edges, smooth color and shading which present significant challenges for texture-descriptor-based loss functions used in the existing solution. ArtifyGAN uses cartoon pictures and unpaired photographs. In order to deal with the significant stylistic differences between photos and cartoons, two losses that are appropriate for cartoonization are proposed: (1) Content loss that is formulated as a sparse regularization in the high-level feature maps of the VGG network, and (2) an edge-promoting adversarial loss for maintaining sharp edges. Furthermore, compared to current methods, ArtifyGAN is far more successful at training. Experimental results show that ArtifyGAN outperforms state-of-the-art models like AnimeGAN, StyleGAN-2 and JOJO-GAN and needs less training time than existing approaches. It may produce high-quality cartoon images while maintaining the structure of the real image.**

Keywords*— (Cartoon generation, face stylization, smooth face cartoonization, ArtifyGAN, AnimeGAN, StyleGAN-2, JOJO-GAN****)***

I. INTRODUCTION

The usage of face animation has seen a significant growth in popularity in the entertainment and gaming industries where personalized avatars and animated emojis are created [1]. However, the current technologies and existing GAN architectures used for generating animated faces are inefficient as they fail to grasp minute details of the face like the texture of the skin and the shape of eyes, thus creating unrealistic animations. These limitations are addressed by proposing a new-state-of-the-art GAN architecture named ArtifyGAN. that is inspired by the existing CartoonGAN architecture [2]. This GAN architecture is trained effectively for generating animated face images to capture minute details of the facial expressions. Also, the ArtifyGAN is optimized in-order to generate animated faces using reference cartoon images. The ArtifyGAN architecture is created by integrating and modifying different architectures in-order to incorporate the strengths of the existing architectures and achieve high quality and diverse results, thereby pushing the boundaries of the existing animated face image generation techniques.

II. RELATED WORK

Several algorithms for Neural Style Transfer (NST) have been created with the aim of generating unique images that blend the content of one image with the style of another image. In this section, we provide a brief overview of various research studies conducted by other researchers in this field. A paper was published by authors proposing a cartoonGAN, a generative adversarial network to improve the existing methods for transforming real-world photos into high-quality cartoon style images by maintaining unique characteristics of cartoon style. A GAN based approach consists of a) generator network, which is used to map photos to cartoon manifolds, containing down-convolution, 8 residual blocks, and up-convolution. b) discriminator network to differentiate between generated & real cartoon images. For preserving content information and maintaining clear edges, a semantic content loss and an edge-promoting adversarial loss function is introduced. Along with 5402 photos from Flickr used for training, 4573 plus 4212 images are used for Makoto Shinkai & Mamoru Hosoda style model and 3617 plus 2302 images for Miyazaki Hayao & Paprika style model training. Only real-world images are used for testing. The results shows that CartoonGAN produces more accurate and visually pleasing cartoon style images by maintaining clear edges and essential features when compared to other state-of-art methods [2].

A paper on AnimeGAN was published by authors in 2020, who proposed a lightweight generative adversarial network for fast transforming a real-world photo into high-quality anime-style images. AnimeGAN consists of generator and discriminator conventional neural networks. Generator is made up of depth wise separable convolution, standard convolution and inverted residual blocks. Discriminator uses standard convolution. To achieve animation effect and color reliability, three loss functions: grayscale style loss, grayscale adversarial

loss and color reconstruction loss are incorporated. Unpaired 6656 real-world photos along with 1792 images from 'The wind rises' movie for Miyazaki Hayao style model, 1650 images from 'Your name' movie for Makoto Shinkai style model and 1553 images from 'Paprika' Movie for Kon Satoshi style model are used for training data. Only real-world photos are used as testing data. As a result, the model produces the transformed images maintaining specific anime style, color and content, outperforming other methods [3].

A paper was published by authors to introduce JoJoGAN, a technique for learning a style mapper from a single example of a style, with a focus on face photos and making it simple for users to add a desired style to their selected photos. This model involves 4 steps: a) GAN inversion where style code & style parameters are obtained by inverting the reference style image. b) Training sets are formed by pairing style code obtained from applying StyleGAN's mixing mechanic and reference style image. c) Fine Tuning StyleGAN to obtain a modified set of parameters that captures desired style. d) Inference to obtained stylized output after applying above 3 steps. Pretrained StyleGAN2 model and StyleGAN's style-mixing property are used along with a single example of the style for training. As a result, this model produces high-quality stylized images preserving identity and desired style. Some limitations include using a single example for style as it may not capture full complexity of style and dependence on pretrained StyleGAN model may limit the range of style that can be effectively captured [4].

In 2021, a paper was published by authors. Proposes a method for unsupervised I2I translation to generate cartoon faces, by optimizing a styleGAN2 pretrained model. Methods involve 2 components: a) FreezeSG, where initial blocks of the generator are freezed so that input image is stimulated to follow the structure of source image. b) Structure loss function to maintain the structure of source image by calculating the MSE between RGB outputs of source and target generator. Flickr-Faces-HQ dataset, which contains HQ images of human faces are used for source domain datasets and for target domain datasets, images from Naver Webtoon, Metfaces and Disney are used. Results show this method outperforms baseline approaches and generates HQ images by maintaining structure of source domain [5].

A paper published in May 2020 by authors, applies GANs to generate cartoon characters from real-world images. The GANs model consists of a Generator network which generates synthetic cartoon images by taking random noise as input and a Discriminator network to differentiate between real and generated images. CaartoonSet100k dataset is used for training the model which contains 100,000 cartoon images. As a result, using GANs creates excellent cartoon style pictures from real-world photos and claims to produce artistic style, smooth shading, and legible edges. The model further can be improved by using larger datasets for training [6].

III. METHODOLOGY

A. Dataset Description

ArtifyGAN is trained on a combination of COCO, CelebA and cartoon dataset. COCO dataset is a visual dataset created by Microsoft and was downloaded from the official website of cocodataset.org [7]. It consists of variety of real-world images which helps the model in understanding and identifying the objects and the context of the images. It consists of more than 330,000 images with 80 different object categories and 5 captions that describe the scene. Fig. 1(a) shows a sample of image from COCO dataset. ArtifyGAN is also trained on CelebFaces Attributes Dataset (CelebA) which consists of more than 200K celebrity images where each image is annotated describing attributes like gender, presence of glass, facial landmarks, etc. This dataset is downloaded from http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html [8]. This dataset is used for training the model to learn the facial expressions and features of human faces for creating realistic animated images for them. Fig. 1(b) shows a sample image from the CelebA dataset. These images are resized to (256,256) before passing through the architecture for training. In-order to capture the style and unique features from the reference cartoon images, ArtifyGAN is trained on cartoon images that are scraped from the safebooru website [9]. These cartoon images as shown in Fig. 1(c) depict certain style, expressive facial expressions, vibrant colors, etc. which will be learnt by the model for better understanding the visual features defining a cartoon style. The dataset was cleaned to remove irrelevant images. They were resized to a fixed size of (256,256) to ensure consistency during training and to ensure that the size is compatible with the architecture. Image pre-processing was done which involved detecting the edge using the skimage library, followed by adding noise in-order to smoothen the images, thereby creating more appealing stylized cartoonized images.

(a) COCO image (b) CelebA image (c) Cartoon image

Fig. 1 Sample images from COCO, CelebA, safebooru dataset

B. Architecture

The proposed model aims to generate cartoon images for real photos and is inspired by CartoonGAN. The GAN (generative adversarial network) framework consists of two Convolutional networks. First, is the Generator(G) network whose goal is to produce output that can fool the Discriminator(D) network. On the other hand, discriminator aims to detect whether the image is a synthesized version or

belongs to a real dataset. In this project, Generator and Discriminator was designed to produce the cartoon images for real dataset by preserving the structure of the real image. In other words, Generator consisted of three main parts i.e. encoder which encodes the features of the images, transformer which transformed the image to match the target manifold and decoder which takes the transformed image as input and produces the cartoon version for real image. Discriminator was a convolution network with a sigmoid layer in the output to detect whether the image belongs to a real dataset or a fake dataset and provides adversarial loss for G. Main objective was to optimize the loss function for the generator and solve the min-max problem as shown below.

$$(G, D) = arg\ min_G max_D L(G, D) \quad (1)$$

As shown in Fig. 6, the Generator network takes the image of shape 256 x 256 with three channels and passes it through a flat convolution stage of kernel size 7 x7 (filters = 64, stride= 1 and padding= 3) which extracts the low-level features and encodes the image. This is the encoder of the Generator architecture. Next phase consists of two down convolution blocks to further compress and encode the image i.e. reduce the height H and weight W by a factor of 2 by using kernel size 3x3. First down convolution block had 128 filters with stride 2 and padding 1 whereas the second convolution block had 256 filters with stride 2 and padding 1. This down sampling technique is used to reduce the spatial dimension of the feature maps and to capture local signals in the images. After down sampling the image, the encoded image passes through 10 residual blocks (kernel size = 3, filters = 256, stride = 1 and padding = 1) and this stage is the transformer of the network. They capture both the content features i.e., the structure and information of the image and the manifold features to represent the specific style and characteristic of cartoon images. By repeatedly applying the network, the model was able to preserve the content of the real image and as well as transform the image into cartoon style by adjusting the features at each block. In this phase, two loss functions play a key role in learning the optimal weights for each layer during training. First is the adversarial loss which tells what extent the Generator must output images to match the target manifold and this information is received by the Discriminator network. Second is the content loss, which preserves the images structure and is defined using the VGG-16 pretrained model. The VGG-16 model is used to extract features from the synthesized images and real images and the loss is defined as the L1 norm of the difference between the two as shown below.

$$L(content)G = L1\left[\left(VGG\left(G(i = image)\right) - VGG(Real\ Image)\right)\right] \quad (2)$$

As the cartoon images have clear edges and smooth shading, there might be style differences which result in change in the representation and characteristics. Hence, to avoid these changes L1 regularization is used which performs better than L2 norm. Hence, total loss is defined as a combination of content and adversarial loss. “w” is the hyper-parameter which is used to tell how much content from the real image has to be preserved. Higher value indicates more content must be preserved. Equation is shown below.

$$L(G, D) = L_{avd}(G, D) + [w] * L_{cont}(G, D) \quad (3)$$

Hence, by balancing both content as well as adversarial loss, the model was able to map input images to cartoon style by minimizing the loss function L(G,D). In the last phase, the decoder stage consisted of two up-convolution blocks by using kernel size 3x3 to restore the original dimension of the image. First up-convolutional block has 128 filters with stride 2 and padding 1 whereas the second up-convolutional block has 64 filters with stride 2 and padding1. Last convolution layer was used to produce an output image with three channels by using kernel size 7x7 and three filters. Also, sigmoid activation function was used so that pixel values range from 0 to 1. Each convolution layer was followed by batch normalization and ReLU activation function.

As shown in Fig. 6, the discriminator network is a shallow network as compared to Generator architecture. Main goal of the discriminator was to tell whether the image was synthesized from G or belongs to a real cartoon dataset. As clear edges are present in real cartoon data and by passing only synthesized images and cartoon images, the discriminator will clearly detect the edges and correctly classify each image. Hence, to fool the discriminator edge smoothing was done using edge detection tool from python skimage library and gaussian noise was added to smooth the edges as shown below in Fig. 2.

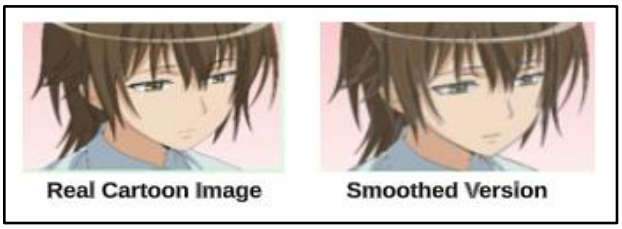

Fig. 2 Real and Smooth Cartoon Image

Therefore, the aim of the discriminator is to maximize the probability to clearly map real cartoon images as real and map smooth cartoon images and synthesize images as fake. Initially the mix of images (synthesized, real and smooth) are passed through flat layers which consist of a convolutional layer with kernel size 3 x 3, 32 filters, stride 1 and padding 1. This layer captures the features and encodes the images. After this, down sampling was done by using two down convolution blocks with stride 2 to further reduce the dimension and capture the local information for classification task. In this stage, important information was captured while unnecessary details were discarded. In the next stage, a feature construction block was applied to further process the feature maps. This block consists of one convolutional layer with kernel size 3x3 and the last layer is the output layer which gives the classification score to which class the images belong to real or fake cartoon. This feedback was provided to the generator network in terms of adversarial loss to produce more visually appealing cartoon images by minimizing the loss function. So based on

adversarial loss, the generator optimizes its weights to produce desired output. Below is the equation for adversarial loss.

$$L_{adv}(G,D) = S_{realcartoon}[log(D)] + S_{smoothcartoon}[log(1-D)] + S_{Gimage}[log(1-D)] \quad (4)$$

Each convolution layer was followed by batch normalization and leaky ReLU at alpha = 0.2. The original cartoon Gan has eight residual blocks and two more convolution blocks after the residual blocks. In the ArtifyGAN, residual blocks have been increased from 8 to 10 to capture the content of the real image more effectively in a lesser number of epochs. Also, removing the extra convolution block helped to improve the efficiency of the model by reducing the computation time. ArtifyGAN architecture is shown in Fig. 6. Hence, with this proposed model optimal and efficient results were achieved in lesser number of epochs i.e. at 79 epochs.

C. Experiment Setting

The proposed model was trained three different times by implementing different hyper-parameters such as varying the number of residual blocks, changing the kernel size, increasing and decreasing the "w" which is used to preserve content loss, and changing learning rate. The CartoonGAN was trained on flickr data and cartoon images were taken from animated movie "spirited away" screenshots. In ArtifyGAN, the real images dataset was obtained by combining coco and celeb dataset whereas cartoon images were obtained from safebooru. Below table shows the different hyper-parameters taken for each round.

TABLE I. HYPERPARAMETERS FOR EACH ROUND

Training Round	***Residual Blocks***	***First layer Kernel Size***	***Learning Rate***	***w (Content Loss)***	***Batch Size***
Round 1	8	7x7	0.005	10	16
Round 2	10	3x3	0.1 (exponential decay)	0	16
Round 3	10	7x7	0.001	0.00001	16

For the first round, the model was trained for 240 epochs whereas for other rounds the model was trained for 210 epochs. All the experiments were performed with the identical setup to maintain consistency in the model evaluation process. In training round 1 after the 240 epochs the content loss was preserved but the output images don't resemble the cartoon styles as shown in Fig. 3. This happened due to the large value set for "w", the model preserved the context of the image, but adversarial loss was very high.

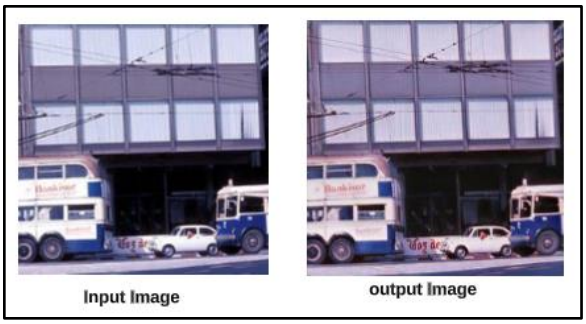

Fig. 6 Input and output image in round 1

In the second round, the value for "w" was reduced from 10 to 0 and to preserve the content and structure of the image, the number of residual blocks was increased from 8 to 10. Also, the kernel size was changed from 7x7 to 3x3, and exponential learning rate was used. Fig. 4 shows the output after 210 epochs. After training the model, the model was able to produce comic style images, but the content of the image was not preserved.

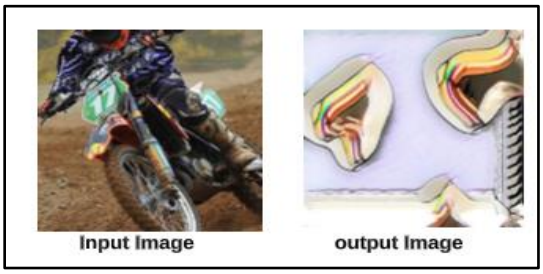

Fig. 4 Input and output image in round 2

In the third round, the value for "w" was set at 0.00001 with 10 residual blocks. Learning rate was set at 0.001 and kernel size 7x7 was taken. In this round, the model was able to produce comic style images from 70 epoch onwards i.e., the model converges faster as compared to other previous rounds and original CartoonGAN. With these hyper-parameter settings, the model achieved optimal results in lesser number of iterations thereby saving computation resources. Fig. 5 shows the output from the model. In this the model was able to minimize both content as well as adversarial loss.

Fig. 5 Input and output image in round 3

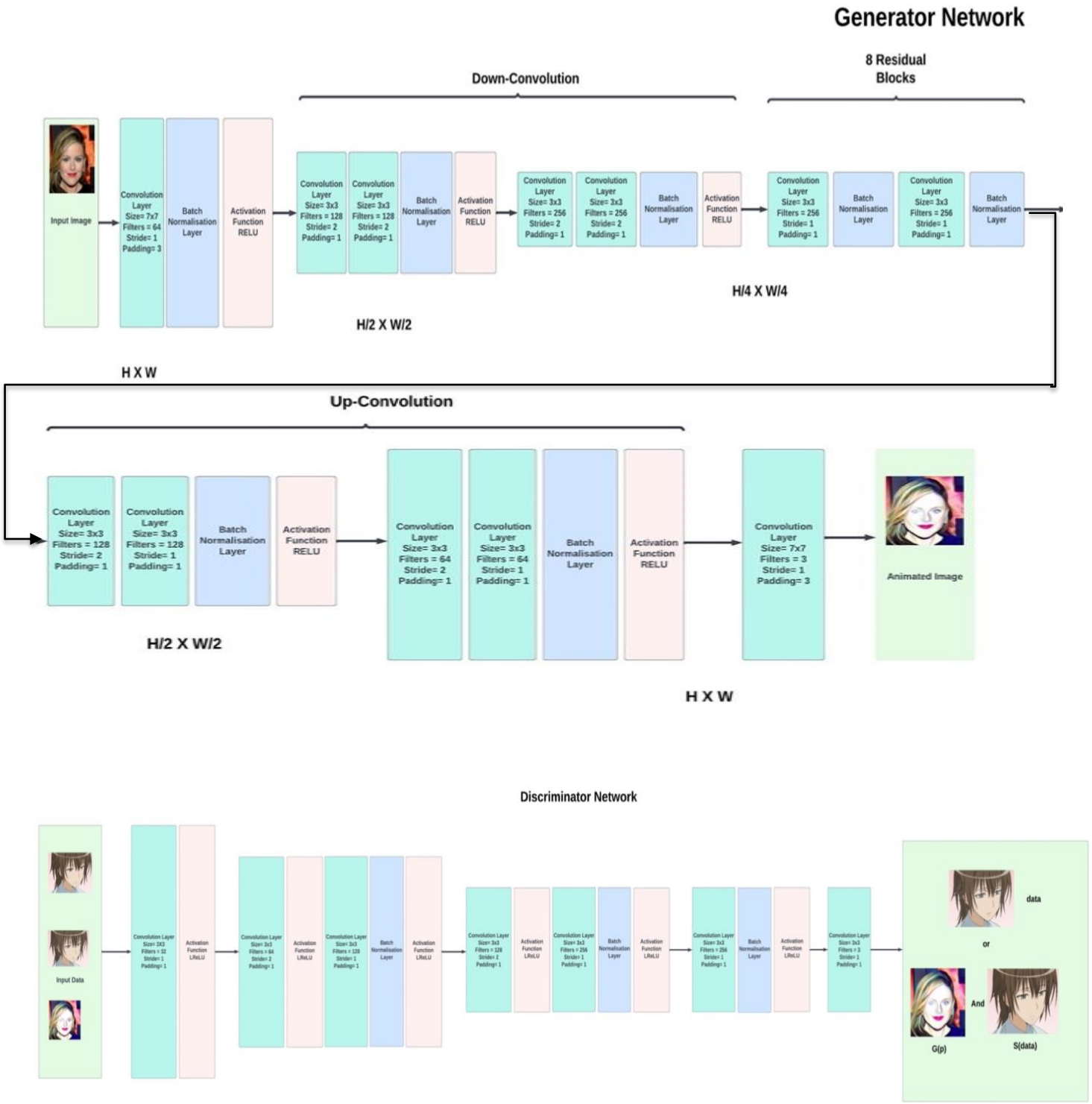

Fig. 6 ArtifyGAN Generator and Discriminator Architecture

All these experiments were carried on GPU (Google Colab) using Jupyter Notebook and the model was created using python3 version and libraries PyTorch, NumPy, Matplotlib, PIL, OpenCV, tqdm, TensorBoard. Table II shows the hardware and software requirements.

TABLE II. HARDWARE AND SOFTWARE REQUIREMENTS

Category	***Requirement***
Hardware	GPU (Google Colab)
Software	Google Colab (Jupyter notebook)
Libraries	PyTorch, NumPy, Matplotlib, PIL, OpenCV, tqdm, TensorBoard

IV. RESULT AND ANALYSIS

A. Quantitative Results

Fig.7 illustrates the content loss of the model which is decreasing at the beginning and then increases after 70 epochs whereas Fig. 8 illustrates the adversarial loss of ArtifyGAN. During the training, the loss is increasing for the first few epochs and then starts decreasing constantly after 30 epochs. The content loss directs the generator to create samples that closely reflect the original data in terms of content by comparing feature representations collected from both the real and created samples, but we must find a good balance between adversarial and content loss. It can be seen in Fig. 7 and Fig. 8 that the loss is stable and balanced around 74 epochs.

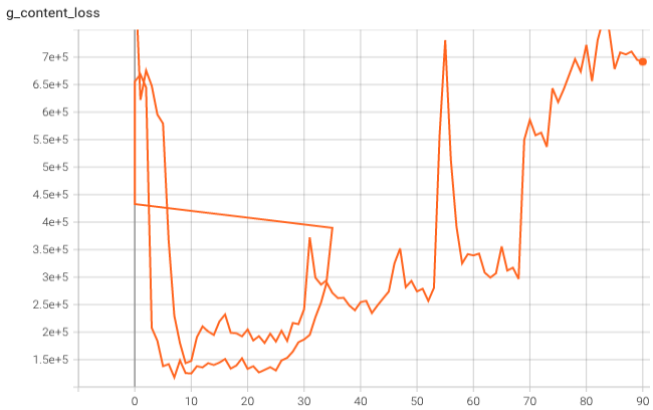

Fig. 7. Content loss of ArtifyGAN with 90 epochs.

Fig. 8. Adversarial loss of ArtifyGAN with 90 epochs

B. Qualitative Results

We compare ArtifyGAN with three recently proposed methods in CNN-based stylization, namely JojoGAN, StyleGAN-2 and AnimeGAN. For a fair comparison, we provided the same image as the original image. Fig. 9 illustrates the result of all the given methods whereas ArtifyGAN produces the best result and preserves the structure of the original image whereas we can see JojoGAN, StyleGAN2, and AnimeGAN does not produce satisfactory result and modifying the original image structure.

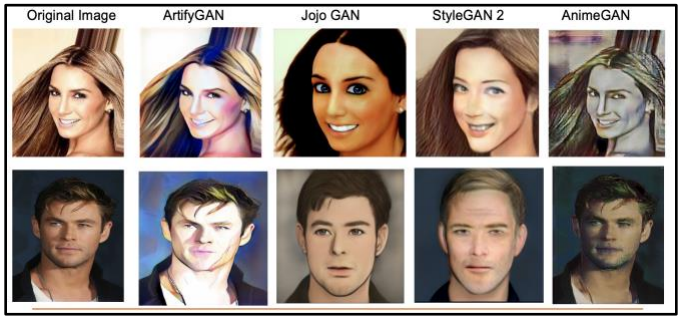

Fig. 9. Comparison of ArtifyGAN with other models

C. Observations

By increasing the residual blocks from 8 to 10 and taking w = 0.00001, the model was able to learn the content of the real image in lesser number of epochs. As we can see in Fig 7 after 78 epochs, the content loss started increasing whereas adversarial loss, this shows that by training the model for larger epochs model forget the content of the real image. Hence right balance between content and adversarial is important which was achieved in the 79th epoch.

ArtifyGAN improves its computational time by increasing the residual blocks and removing extra convolution layers after residual blocks. Also, the inference time is too less compared to existing solution. Generator model with optimal weights can give a cartoon image in less than 2 seconds. On the contrary, the model was trained on COCO and CelebA dataset, we could have trained on more diverse data with high number of epochs to get better quality output.

V. CONCLUSION

ArtifyGAN, our model has made significant contributions to generating high-quality animated images. By designing the model from scratch, we have achieved more control and customization, resulting in impressive animations despite limited data and training time. This highlights the importance of our work in providing an efficient and effective alternative to existing complex models. Our model reduces resource requirements and enables real-time applications, benefiting entertainment, gaming, and virtual reality industries. ArtifyGAN's flexibility and customization offers better tailored results and advance the field of animated image generation. Future research for ArtifyGAN includes exploring expression transfer, facial feature modification, training on a larger and diverse dataset, and age progression. These enhancements would increase ArtifyGAN's ability to transfer expressions between faces, modify facial features beyond expressions, improve its generalization capabilities, and generate realistic age progression effects. These developments would increase ArtifyGAN's ability to produce a wider variety of animated images and provide users with more creative options.

REFERENCES

[1] A. Pumarola, A. Agudo, A. M. Martinez, A. Sanfeliu, and F. Moreno-Noguer, "GANimation: One-Shot Anatomically Consistent Facial Animation," *International Journal of Computer Vision*, Aug. 2019, doi: https://doi.org/10.1007/s11263-019-01210-3.

[2] Y. Chen, Y.-K. Lai, and Y.-J. Liu, "Cartoongan: Generative Adversarial Networks for photo cartoonization," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. doi:10.1109/cvpr.2018.00986 .

[3] J. Chen, G. Liu, and X. Chen, "AnimeGAN: A novel lightweight gan for photo animation," *Communications in Computer and Information Science*, pp. 242–256, 2020. doi:10.1007/978-981-15-5577-0_18

[4] M. J. Chong and D. Forsyth, "Jojogan: One shot face stylization," *Lecture Notes in Computer Science*, pp. 128–152, 2022. doi:10.1007/978-3-031-19787-1_8 .

[5] J. Back, "Fine-Tuning StyleGAN2 For Cartoon Face Generation," arXiv (Cornell University), Jun. 2021, doi: 10.48550/arxiv.2106.12445.

[6] G. Guruprasad*, G. Gakhar, and D. Vanusha, "Cartoon character generation using generative Adversarial Network," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 9, no. 1, pp. 1–4, 2020. doi:10.35940/ijrte.f7639.059120

[7] "COCO - Common Objects in Context," *cocodataset.org*. https://cocodataset.org/

[8] "Large-scale CelebFaces Attributes (CelebA) Dataset," *mmlab.ie.cuhk.edu.hk*. http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

[9] "Safebooru - Anime picture search engine!," *safebooru.org*. https://safebooru.org/