# FrameSequenceGAN: Recurrent Animated Video Generation Using Generative Adversarial Networks and Sequence Discriminators

Chan Woo Kim
ck15@williams.edu
ck15@williams.edu
Williams College
Williamstown, Massachusetts



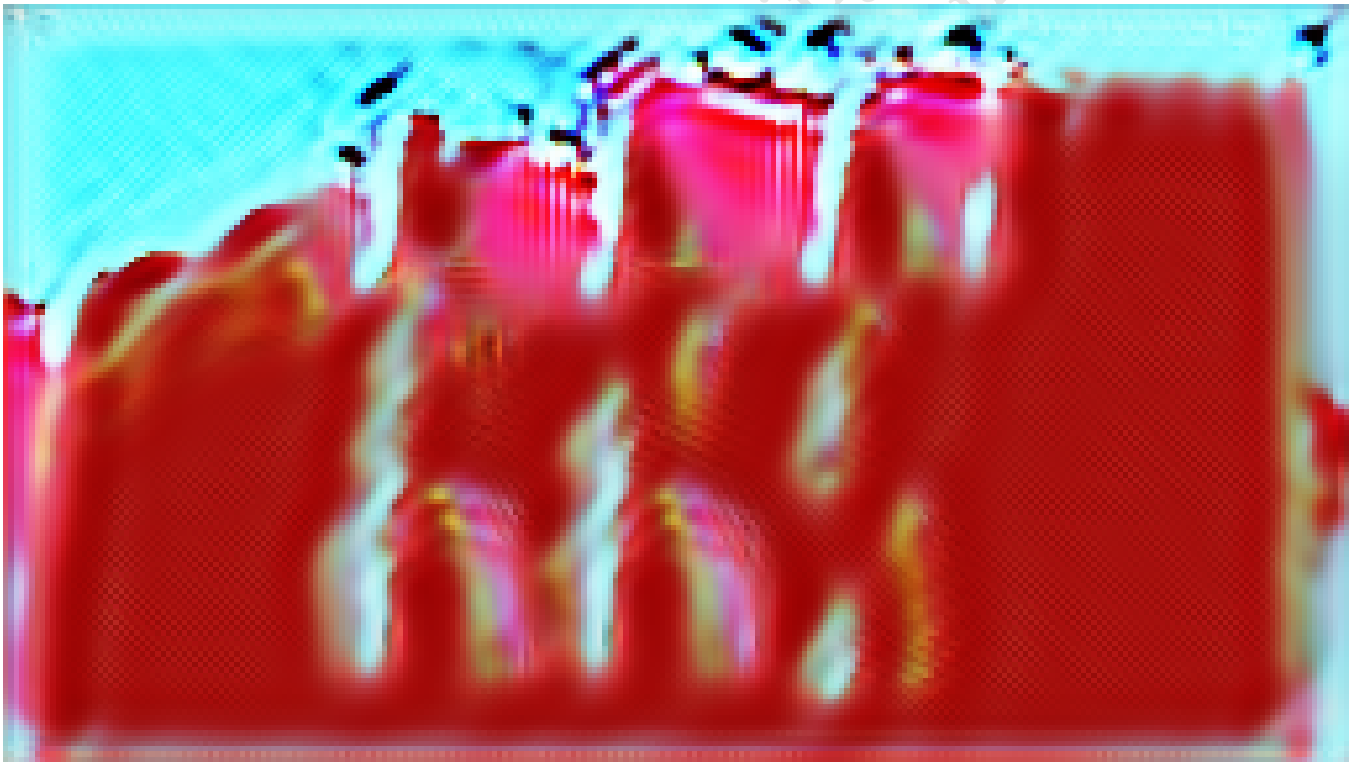

**Figure 1: A Frame From an Animation Generated by FrameSequenceGAN.**

## ABSTRACT

Generative Adversarial Networks have shown immense success in the art space, through its ability to generate seemingly original artworks through learning from existing art. This paper proposes a model that applies such technology to produce art in a sequence, to rather form a sequential artistic *animation*. Specifically, the FrameSequenceGAN is a recurrent system that takes as input a previous frame of an animation and predicts the next

frame as output. It trains against two discriminators, a standard discriminator in adversarial training that judges how closely the output resembles the dataset, and a sequence discriminator that judges how well the output forms a sequence with its input.

# 1 INTRODUCTION

The ability to automatically create arbitrarily many animations given one animation would enhance an artist's creative work. Such a model would generate new movements and shapes of objects that are still grounded in the original animation. The goal is that such a tool would inspire an artist to explore new ideas they may have not initially thought of until the model began producing outputs resembling them, or even use these outputs in the final animation.

In order to achieve this, we utilize Generative Adversarial Networks (GANs), which have proven to be one of the most promising candidates in image generation tasks. The idea is to utilize the same notion of adversarial training that brings about incredible outputs that impressively resemble a given data-set, and apply another instance of adversarial training for *sequential* generation.

We propose a FrameSequenceGAN, which ultimately trains a generator that receives as input some frame in the animation and produces a frame of the next time-step.

# 2 LITERATURE REVIEW

## 2.1 Generative Adversarial Networks (GAN)

GANs were introduced by Goodfellow *et. al.*, [2], which use a *adversarial nets* framework whereby a generative model is "pitted" against an adversary. The adversary is usually referred to as the "discriminator," which learns to judge whether a given sample is from the true data-set or a "decoy" produced by the generator to mimic the true data-set. The goal is for the generator and the discriminator to improve concurrently until an equilibrium is reached and the outputs of the generator are indistinguishable from those from the true data distribution. A standard indicator for this is when the discriminator chooses between the "decoy" and true data with 50% chance.

## 2.2 GAN-Based Video Generation

Few of the most direct adaptations of GAN for video generation are proposed in [6] and [8], which simply adds a new temporal dimension to the samples. Instead of 2D convolutions on 2D images, they use 3D convolutions on space-time cuboids where the volume of the cuboids is the number of generated animated frames. The inherent limitation to this approach is that it is fixed-length, since the size of the cuboid is permanently fixed from model initialization.

Many new approaches to GAN-based video generation has been proposed in specific domains of video generation, such as facial synthesis. Pumarola *et. al.* [5] utilized Conditional GANs [4] based on AUs (Action Units) to generate anatomically-aware facial animations. By implementing the GAN to be conditioned on a vector that indicates the magnitude of each action unit, by interpolating on what is a continuous domain, an animation could be generated. Vougioukas *et. al.* [9] proposed a temporal generative adversarial network that takes as input an audio signal containing speech and a still image of a face to output a facial animation of mouthing the words. This model utilizes two encoders, a decoder, and a Recurrent Neural Network (RNN) to generate the sequential frames, conditioned upon the audio signal. A frame discriminator pushes the generator to produce realistic frames, and a sequence discriminator uses a CNN and a Gated Recurrent Unit (GRU) to determine how realistic the sequence of frames is. Similarly, the MoCoGAN architecture proposed in [7] also uses an approach using RNNs, using separate latent spaces for motion and content. It also employs a separate sequence discriminator, which uses 3D convolutions. Hamada *et. al.* [3] proposes the Progressive

Structure-conditional GANs, which was able to produce high resolution anime animations.

# 3 PROPOSED MODEL

The FrameSequenceGAN is divided into three components: a Generator, a Frame Discriminator, and a Sequence Discriminator. The Generator of this model is not a standard generator in a GAN system in that its input is not a random vector sample of a certain distribution, but an image at a certain timestamp. Given an image sample $I_t \in \mathbb{R}^{H \times W \times 3}$ at time-step $t$, we want to train a Generator $G(I_t)$ such that it predicts $I_{t+1}$. Specifically, $G$ is a Convolutional Neural Network that aims to *model after* the transformation: $T : I_t \longrightarrow I_{t+1}$.

The Frame Discriminator is the standard discriminator in a GAN model. It aims to train its weights such that it can make the following judgements: $\widehat{I_{t+1}} \nsim p_{data}$ and that $I_{t+1} \sim p_{data}$, where $p_{data}$ is the original dataset distribution and $\widehat{I_{t+1}}$ is a predicted frame made by the Generator. In other words, it tries to judge whether a given image frame is generated by the Generator or if it exists in the true dataset. With the Generator pitted against the Frame Discriminator, we aim to train the Generator such that it generates frames that are realistic to the dataset.

However, for the purpose of video generation, there exists additional challenges for the Generator than to simply generate realistic frames that fit in the true dataset distribution: the Generator must generate frames $G(I_t) = \widehat{I_{t+1}}$ such that the transformation from $I_t$ to $\widehat{I_{t+1}}$ constitutes a realistic sequence of two frames. Even if $\widehat{I_{t+1}}$ is a realistic image that fits in the true dataset, it is of no use in animated video generation if it has completely no resemblance to $I_t$, since an animation is a reasonably smooth *sequence* of frames, not just a mix of random images.

To solve the above challenge, we add the Sequence Discriminator to the model. The Sequence Discriminator is unique to the FrameSequenceGAN and it aims to update its weights such that it can differentiate between the following difference maps: $(\widehat{I_{t+1}} - I_t)$ and $(I_{t+1} - I_t)$. In other words, it assumes that there is a distinguishable distribution $Z$ defined by $(I_{t+1} - I_t) \in Z$ for all $t$, and attempts to distinguish whether a given difference map belongs in Z. The purpose of this discriminator is simply in asking whether the transformation from one frame to another produced by the Generator is a transformation that can be observed in the original animation. Through having the Generator pitted against the Sequence Discriminator, we aim to train the Generator such that it moves closer to producing such transformations: $(I_t \longrightarrow \widehat{I_{t+1}}) \sim p_{data}$.

Instead of a joint optimization process, the generator updates its weights separately twice, once against the Frame Discriminator and once against the Sequence Discriminator.

## 3.1 Generator & Frame Discriminator

As explained previously, the Generator $G(I_t)$ is pitted against a Discriminator $D$ to adversarially train against it. We employ the most traditional form of this game, not the newer frameworks such as Wasserstein GAN proposed inx [1] that improves upon the instability of traditional GANs. Specifically, the minimax game between the generator and the discriminator can be represented as

$$\min_{\theta G} \max_{\theta D} \mathbb{E}_{I_{t+1} \sim p_{data}}[\log D(I_{t+1})] + \mathbb{E}_{I_t \sim p_{data}}[\log(1 - D(G(I_t)))], \quad (1)$$

where $\theta G$ and $\theta D$ are parameters of the generator and the discriminator, respectively.

## 3.2 Sequence Discriminator

The Generator and Sequence Discriminator $S(d)$ also engage in adversarial training. The input to the Sequence Discriminator is a difference map between two images. Similar to the Generator and the Frame Discriminator, they engage in a separate

minimax game represented as follows:

$$\min_{\theta G} \max_{\theta S} \mathbb{E}_{I_t, I_{t+1} \sim p_{data}}[\log S(I_{t+1} - I_t)] + \mathbb{E}_{I_t \sim p_{data}}[\log(1 - S(G(I_t) - I_t))]. \quad (2)$$

# 4 IMPLEMENTATION DETAILS

This implementation used TensorFlow 2.0 Keras API and one K80 GPU on an AWS EC2 instance to train.

The dimensions of the images are $144 \times 256 \times 3$, where the third dimension refers to the RGB channel. All the images were initially encoded in 8-bit RGB then linearly scaled to $(-1, 1)$. The output of the generator is also within the range $(-1, 1)$, which is specifically why the *tanh* activation was used.

The Generator, Frame Discriminator, and Sequence Discriminator model architectures can be found in **Table 1**, **Table 2**, and **Table 3** respectively. CNN kernels were initialized through randomly sampling from a uniform distribution.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Conv2D | (144, 256, 64) | 9472 |
| LeakyReLu | (144, 256, 64) | 0 |
| Dropout | (144, 256, 64) | 0 |
| Conv2D | (144, 256, 128) | 204928 |
| BatchNormalization | (144, 256, 128) | 512 |
| LeakyReLu | (144, 256, 128) | 0 |
| Dropout | (144, 256, 128) | 0 |
| Conv2D | (144, 256, 256) | 819456 |
| BatchNormalization | (144, 256, 256) | 1024 |
| LeakyReLu | (144, 256, 256) | 0 |
| Dropout | (144, 256, 256) | 0 |
| Conv2D | (144, 256, 128) | 819328 |
| BatchNormalization | (144, 256, 128) | 512 |
| LeakyReLu | (144, 256, 128) | 0 |
| Dropout | (144, 256, 128) | 0 |
| Conv2D | (144, 256, 64) | 73792 |
| BatchNormalization | (144, 256, 64) | 256 |
| LeakyReLu | (144, 256, 64) | 0 |
| Dropout | (144, 256, 64) | 0 |
| Conv2D | (144, 256, 3) | 1731 |
| tanh activation | (144, 256, 3) | 0 |

**Table 1: Generator Model Architecture**

# 5 RESULTS

# 6 ACKNOWLEDGMENTS

Thanks to Dr. Andrea Danyluk at Williams College for advising this research and NYC artist Joshua Frankel for providing an opportunity to explore this field of research and a platform to display this work. Model outputs from this research were directly put on display for the GRAND BAND musical ensemble at Peak Performances @ Montclair State University as part of Joshua's wider animated work.

# REFERENCES

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875* (2017).

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[3] Koichi Hamada, Kentaro Tachibana, Tianqi Li, Hiroto Honda, and Yusuke Uchida. 2018. Full-body high-resolution anime generation with progressive structure-conditional generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 0–0.

[4] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[5] Albert Pumarola, Antonio Agudo, Aleix M Martinez, Alberto Sanfeliu, and Francesc Moreno-Noguer. 2018. Ganimation: Anatomically-aware facial animation from a single image. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 818–833.

[6] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. 2017. Temporal generative adversarial nets with singular value clipping. In *Proceedings of the IEEE International Conference on Computer Vision*. 2830–2839.

[7] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. 2018. Mocogan: Decomposing motion and content

| Layer (type) | Output Shape | Param # |
|---|---|---|
| INPUT | (144,256, 3) | |
| Conv2D | (72, 128, 64) | 4864 |
| LeakyReLu | (72, 128, 64) | 0 |
| Dropout | (72, 128, 64) | 0 |
| Conv2D | (36, 64, 128) | 204928 |
| BatchNormalization | (36, 64, 128) | 512 |
| LeakyReLu | (36, 64, 128) | 0 |
| Dropout | (36, 64, 128) | 0 |
| Conv2D | (36, 64, 32) | 102432 |
| BatchNormalization | (36, 64, 32) | 128 |
| LeakyReLu | (36, 64, 32) | 0 |
| Dropout | (36, 64, 32) | 0 |
| Conv2D | (36, 64, 64) | 18496 |
| BatchNormalization | (36, 64, 64) | 128 |
| LeakyReLu | (36, 64, 64) | 0 |
| Dropout | (36, 64, 64) | 0 |
| Conv2D | (36, 64, 32) | 18464 |
| BatchNormalization | (36, 64, 32) | 128 |
| LeakyReLu | (36, 64, 32) | 0 |
| Dropout | (36, 64, 32) | 0 |
| Flatten | (73728) | 0 |
| Dense | (1) | 73729 |

**Table 3: Sequence Discriminator Model Architecture**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| INPUT | (144,256, 3) | |
| Conv2D | (72, 128, 64) | 4864 |
| LeakyReLu | (72, 128, 64) | 0 |
| Dropout | (72, 128, 64) | 0 |
| Conv2D | (36, 64, 128) | 204928 |
| BatchNormalization | (36, 64, 128) | 512 |
| LeakyReLu | (36, 64, 128) | 0 |
| Dropout | (36, 64, 128) | 0 |
| Conv2D | (36, 64, 128) | 147584 |
| BatchNormalization | (36, 64, 128) | 512 |
| LeakyReLu | (36, 64, 128) | 0 |
| Dropout | (36, 64, 128) | 0 |
| Conv2D | (36, 64, 128) | 147584 |
| LeakyReLu | (36, 64, 128) | 0 |
| Dropout | (36, 64, 128) | 0 |
| Conv2D | (36, 64, 32) | 102432 |
| BatchNormalization | (36, 64, 32) | 128 |
| LeakyReLu | (36, 64, 32) | 0 |
| Dropout | (36, 64, 32) | 0 |
| Flatten | (73728) | 0 |
| Dense | (1) | 73729 |

**Table 2: Frame Discriminator Model Architecture**



for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1526–1535.

[8] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. 2016. Generating videos with scene dynamics. In *Advances in neural information processing systems*. 613–621.

[9] Konstantinos Vougioukas, Stavros Petridis, and Maja Pantic. 2018. End-to-end speech-driven facial animation with temporal gans. *arXiv preprint arXiv:1805.09313* (2018).