# Frame Interpolation by Extending SepConv with Generative Adversarial Networks

Matthew Yang
mjyang@g.clemson.edu

Clemson University
December 6, 2024

## Abstract

Frame interpolation using Generative Adversarial Networks is not a recent advancement in this technology, but the potential of extending previous works could be further explored. Frame interpolation of two images with a large content distance may be an issue with a simple network. In an attempt to address this issue, a network previously proposed to interpolate images is adapted and fitted into a GAN style architecture. By utilizing both adversarial learning and an encoder-decoder generation network, this study aims to keep both simplicity in the implementation while achieving an interpolation result. Two different methods of using the network to generate a frame frame were also investigated. A user study was conducted to investigate whether the result of the interpolation is received well by the average viewer. Using three different implementations of the same architecture style, the results were unexpected, and multiple problems beyond possible control arose. The project website is available at https://needsaltforfries.github.io/ai_interpolator.github.io/.

## 1 Introduction

Frame Interpolation is the process of taking two images and generating an image such that the resulting frame represents a transition between the two inputs. Frame Interpolation has a wide range of applications, including video file compression [12] and creating visually-pleasing effects via slow motion [13]. The challenge of creating that in-between frame lies in synthetically creating data that does not exist [4]. With the advent of Artificial Intelligence, interpolating between frames can be done without access to the same tools as the original artists. 3D and hand-drawn software can be artificially extended using AI to produce and edit videos. Developments in AI allows for the generated frame to be created by feeding the data from the two images into the network [7]. The use of Generative Adversarial Networks in prior research has been known to produce high-quality image results. By extending this concept to frame interpolation, the prospect of achieving high quality frame interpolation appears feasible.

Frame interpolation has been achieved using AI networks by using more complicated methods, such as heat maps or motion vectors to distort the content of the image into a suitable form in the generated frame [3, 11]. These methods appear costly in both time and computational power. The detrimental aspects of these methods arise another challenge of this implementation. The goal is to find a simple method to AI frame interpolation using an adversarial network while producing acceptable generation results. In order to achieve a simple network that outputs a favorable result, multiple components must be used in tandem. The proposed implementation in this paper is to utilize adaptive separable convolution as the image generator, and a typical Convolutional Neural Network for the image discriminator [9]. By leveraging adversarial learning, the generator network, in theory, should produce more realistic results. By appending the generated frames to the original 60 frames per second video, the original video can be converted into a 120 fps video.

## 2 Related Works

The Deep Convolutional Generative Adversarial Network architecture is comprised of two networks, a generator and a discriminator. The generator aims to produce an output that can fool the discriminator, while the discriminator determines if an image is either fake or real. [5] Goodfellow et al. form the adversarial loss problem as

$$V(G,D) = E_{x-p_{data}}[logD(x)]+E_{x-p_g}[log(1-D(x)]$$

where training the GAN becomes a minmax problem

$$maxminV(G, D)$$

The result is that the discriminator will maximize the likelihood the image input is real or fake, while the generator will minimize the chances of the discriminator giving the correct output. Advancements in the DCGAN architecture allows for conditions to be attached to the generator and discriminator. [8] Additional information, such as a label or images, can be used for guiding the output of the neural network to an intentional result. For video interpolation, this key change becomes necessary to produce frames close to the desired images.

Training the GAN proves difficult because the networks may become unstable if improperly balanced. By utilizing a Wasserstein-GAN with a Gradient Penalty [6, 1], the networks become more robust against collapsing. The Wasserstein GAN loss function becomes

$$\mathcal{L}_{\text{WGAN}} = \mathbb{E}_{x\sim P_{\text{data}}}[D(x)] - \mathbb{E}_{x\sim P_{\text{gen}}}[D(G(z))]$$

By taking the mean of the discriminator loss and multiplying it by -1, the output of the image can be further influenced depending on how well the generator is fooling the discriminator. According to Arjovsky et al., the Wasserstein GAN should be able to balance the generator and discriminator during training[1, 2]. Knowledge of Image-to-Image translation useful due to the similarity to interpolation. Isola et al. show that the loss of a CGAN generator can be interpreted as

$$L_{L1}(G) = \mathbb{E}_{x,y,z}[||y - G(x, z)||_1]$$

[7]. This shows that the L1 norm can be used for the generator and have the neural network still act as a Conditional GAN. Therefore, the final objective becomes

$$L_{GAN}(G, D) + \lambda L_{L1}(G)$$

where lambda is a regularization factor. The use of the L1 norm, as opposed to the L2 norm, demonstrated better pixel accuracy when performing detailed image processing [9, 7]. Use of the L2 norm in both experiments performed by both Niklaus and Isola gave less detailed results and blurry artifacts on their images.

Video interpolation using Adaptive Separable Convolution is able to interpolate between entire video frames at once. [9] By using 1-D kernels for full-frame interpolation, higher quality results can be produced. Their architecture takes in two input images then feeds them through an encoder-decoder structure. Using skip layers, the features and colors extracted from the convolutional layers pool into bilinear upsampling layers. Figure 1 abstractly depicts the layout of the generator.

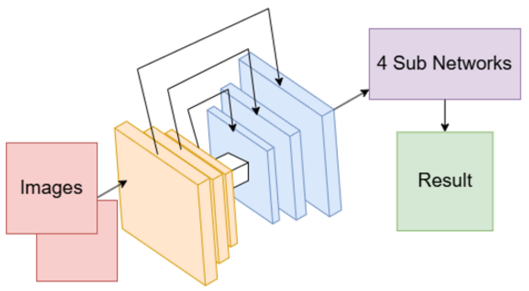

Figure 1: An adaptation of the SepConv architecture. Two images are inputted into an encoder-decoder structure, extracting then reconstructing the features. The encoder is comprised of convolutional layers of 32, 64, 128, 256, and 512 output channels. The decoder is comprised of convolutional layers with 256, 128, 64, 32 output channels. ReLU activation layers are used at the end of each convolutional layer. The output of the main network is sent to four convolutional subnetworks each with a ReLU activation layer. [9].

## 3 Approach

There are two implementations of the video interpolator to determine what could be changed about the frame generation process to yield better results. The first implementation is based off of a DCGAN architecture. The second attempts to recreate a WGAN-GP. The training procedure and network architecture differ between the two implementations, but primarily follow the same method to create the interpolated video. Both implementations take in two images $I_0$ and $I_1$ from a directory as inputs after converting them into tensors using Python Image Libraries. The network then outputs an RGB image $I_{0.5}$ of the interpolation between the two images and stored in another directory. Using moviepy, the original frames and the generated output are interwoven such that each interpolated frame is between the two input images in the video file.

$$I_0, I_{0.5}, I_1$$

### 3.1 Training

The neural network was implemented in Pytorch and visualized using Jupyter Notebook. The training data is taken from the Vimeo90k dataset [14] and the ATD12k dataset[10]. The

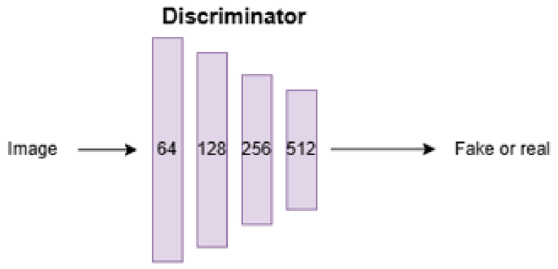

Figure 2: An abstract view of the discriminator. The discriminator is made of convolutional layers with 64, 128, 256, 512 output channels each with a LeakyReLU activation layer. The final layer is a linear layer with a sigmoid activation.

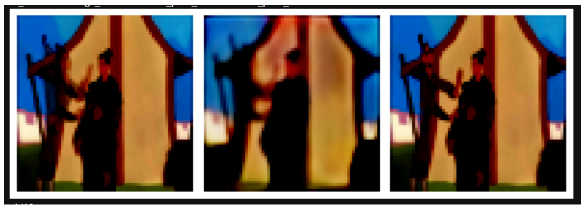

Figure 3: An example output of the generator while training. The pictures are the overlayed inputs, generator output, and the ground truth frame respectively.

Vimeo90k dataset is a large dataset of 448x256 resolution triplets taken from over 15 thousand videos, and the ATD12k dataset contains . The content distance, or how close the content in the image is to the target frame, is very close, which can cause the network to fail with large content distances. To supplement this, the ATD12k dataset, is used in conjunction in order to be more robust against larger content distances. The images from both datasets are resized to 64x64 pixels during the training process. Using the Clemson Palmetto Cluster with an A100 GPU, the network takes a batch of 128 images and trains for an arbitrary number of epochs. The training process requires adjusting the hyper-parameters in-between epochs to achieve better results. The Adam optimizer was used with $\beta = 0.5, 0.999$ and the learning rate was set to 0.0003 for the discriminator and generator. When using the Binary Cross Entropy loss function for the generator, the generator was able to output shapes similar to the ground truth within 5 epochs, but the output quickly discolorized. The loss function for the generator was then changed to using the L1 loss function $\lambda|I_{GT} - I_g|_1$. After making this change, the generator began to generate pictures that share the features of the input images. The regularization parameter $\lambda$ was set to $1 <= \lambda <= 100$ during the early stages of training and $0 < \lambda <= 1$ for later stages of training where color began to appear in the output. The network was trained over the course of 30 epochs, with approximately 400 iterations per epoch. Training results were output into the Jupyter Notebook file every 10 iterations. The discriminator often overpowered the generator, while the generator likely could not improve against the discriminator.

## 3.2 Implementations

The original implementation was mistakenly alternating between upscaling and downscaling the image between convolutional layers. This led to large images unable to be processed due to memory issues. It was only able to process 380 x 120px images. This implementation was discarded and changed to follow the proposed method with the SepConv generator. After overhauling the entire network, the SepConv generator was implemented with skip layers and pixel-wise subnetworks [9]. The discriminator became a network of convolutional layers decreasing in layer size with a final linear layer. After 40 iterations on the first epoch, features began to appear in the output, resembling a blurred, gray-scaled copy of the overlayed inputs. Colors started appearing after 5 epochs, and training results for a 64x64px began to fully resemble the input images between 15-30 epochs.

The network was further refined in an experimental implementation to emulate a Wasserstein GAN. The generator was unmodified for this implementation, but the discriminator had the final Sigmoid activation layer removed to fit the output range from [-1, 1]. Due to time constraints, the vision for this specific implementation was not fully realized.

## 4 Results

While in evaluation mode, the network was able to successfully take in two frames as input and output a single frame. The input frames were resized to a resolution of 1280 x 720, and the output frame was visualized using Jupyter Notebook. In less than a second, the A100 GPU on the Palmetto Cluster was able to output the entire frame. Visualizing the output while the program iterated through the directory of original frames allowed for the ability to watch the entire video as the interpolated frames were being processed. After creating the

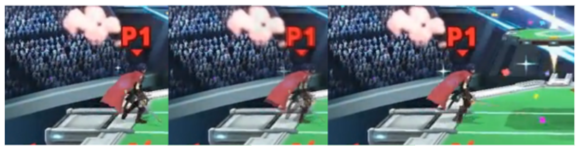

Figure 4: Results of interpolating a 1980x1080 video. The left and right images are the original frames, and the center image is the interpolated result.

Table 1: A user study of 10 people conducted online.

| Preferred Clip | Percentage |
|---|---|
| Control | 50% |
| Edited Clip | 50% |

video, an online, blind user study of 10 people was conducted using Google Forms. Embedding a video into the form required uploading the video to YouTube. Because of how YouTube compresses videos, the 120 frames per second became compressed to a 60 fps video. To compensate, the interpolated and the original 60 fps video was slowed to half-speed and rendered at 60 fps using MAGIX Vegas Pro 16. Table 1 shows the results of the study, reporting that there was likely no percievable difference between the original and the interpolated clip when viewed on the Google Form. Another study of 8 Computer Science students was conducted in person, asking viewers to identify the video clip that appeared smoother. Every viewer identified the interpolated clip as smoother.

## 5 Limitations

At first glance, the video appears smoother, but a frame by frame analysis revealed the weakness of the network. By viewing the video in 0.25x speed, the generated frames appeared to echo the average of the two input images. There was no spatial warping for the generated result. In training, the network appeared to blur the resulting image, but there was little to no blur for the image when the network was set in evaluation mode. To thoroughly test the network, 64x64 patches of the images were fed into the network and reconstructed to make the higher resolution image, which yielded similar results as before. Other weaknesses of this study were introduced because embedding videos on Google Forms drastically reduces the original quality, and requires a link to a YouTube video to embed into the form. The quality of the video could not accurately be captured due to the YouTube encoder. Workaround attempts such as upscaling the video to 2560 x 1440 proved ineffective. Time constraint, including the 12 hour limit also disallowed effective network training. The common 60 fps monitors on most devices also could not accurately display the 120 fps video.

## 6 Future Work

For future revisions of this implementation, the Wasserstein GAN could be implemented to make training more robust. This was attempted late into the study, but time constraints made rewriting the implementation unfeasible. The interpolation results could be further improved by reworking the current discriminator implementation. Unfamiliarity with Pytorch would cease to be an issue, allowing for working more effectively. Training the network could also be switched to using patches of the input images instead of attempting to interpolate the entire frame. The study could be conducted by slowing the video down to quarter speed to compensate for the frame skipping used in the YouTube encoder to get more accurate results.

## 7 Conclusion

This study seeks to explore the Generative Adversarial Network architecture and attempts to implement a GAN by adapting SepConv as a generator. The training process of the network was challenging, due to uncertainty in which factors were implemented incorrectly. The video generator outputs an image more akin to the average of the two inputs rather than an intermediary frame. Despite the poor intermediary frame results, users reported the interpolated video as smoother, but their results could be explained due to random chance or a placebo.

## References

[1] Jonas Adler and Sebastian Lunz. Banach wasserstein gan. *Advances in neural information processing systems*, 31, 2018.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[3] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3703–3712, 2019.

[4] Jiong Dong, Kaoru Ota, and Mianxiong Dong. Video frame interpolation: A comprehensive survey. *ACM Transactions on Multimedia Computing, Communications and Applications*, 19(2s):1–31, 2023.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[6] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[7] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[8] Mehdi Mirza. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[9] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE international conference on computer vision*, pages 261–270, 2017.

[10] Li Siyao, Shiyu Zhao, Weijiang Yu, Wenxiu Sun, Dimitris Metaxas, Chen Change Loy, and Ziwei Liu. Deep animation video interpolation in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6587–6595, 2021.

[11] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018.

[12] Muhammad Usman, Xiangjian He, Kin-Man Lam, Min Xu, Syed Mohsin Matloob Bokhari, and Jinjun Chen. Frame interpolation for cloud-based mobile video streaming. *IEEE Transactions on Multimedia*, 18(5):831–839, 2016.

[13] Joost Van Amersfoort, Wenzhe Shi, Alejandro Acosta, Francisco Massa, Johannes Totz, Zehan Wang, and Jose Caballero. Frame interpolation with multi-scale deep loss functions and generative adversarial networks. *arXiv preprint arXiv:1711.06045*, 2017.

[14] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127:1106–1125, 2019.