2024-05-01

In-between frame generation for 2D animation using generative adversarial networks

Francisco Arriaga Pazos
University of Texas at El Paso

IN-BETWEEN FRAME GENERATION FOR 2D ANIMATION USING GENERATIVE ADVERSARIAL NETWORKS

FRANCISCO ARRIAGA PAZOS

Master's Program in Data and Information Sciences

APPROVED:

Olac Fuentes, Ph.D., Chair

Monika Akbar, Ph.D.

Art Duval, Ph.D.

Stephen L. Crites, Jr., Ph.D.
Dean of the Graduate School

Dedication

to that CHILD

in case you ever forget

IN-BETWEEN FRAME GENERATION FOR 2D ANIMATION USING GENERATIVE ADVERSARIAL NETWORKS

by

FRANCISCO ARRIAGA PAZOS

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Master's Program in Data and Information Sciences

THE UNIVERSITY OF TEXAS AT EL PASO

May 2024

Acknowledgements

To my mother, the wisest person I know, who showed me that in pursuit of the right drive, there is no such a thing as losing time.

To my father, the most tenacious person I know, who showed me that no matter what the starting point might be, putting the work in leads to extraordinary results.

To Carmina, the bravest person I know, who showed me how inspiring it is to see someone rise to the challenge.

To Sara, the strongest person I know, who showed me how illusory limitations are when she got rid of hers by forgetting they were there.

To my advisor, Dr. Olac Fuentes, who guided me through this entire process by distilling for me his vast body of knowledge and experience and fomenting a sense of rigor and focus that has allowed my somewhat unconventional academic interests to come to fruition.

To the Vision and Learning Lab members, who were always down at the trenches with me, sharing practical advice and providing the most intellectually stimulating environment I have ever been a part of.

To Dr. Art Duval, and Dr. Monika Akbar, my committee members, for granting their time and attention to this odd little project of mine.

To you, who have been there from the start and showed me what unconditional truly means.

NOTE: This thesis was submitted to my Supervising Committee on November 30, 2023.

Abstract

Traditional 2D animation remains a largely manual process where each frame in a video is hand-drawn, as no robust algorithmic solutions exist to assist in this process. This project introduces a system that generates intermediate frames in an uncolored 2D animated video sequence using Generative Adversarial Networks (GAN), a deep learning approach widely used for tasks within the creative realm. We treat the task as a frame interpolation problem, and show that adding a GAN dynamic to a system significantly improves the perceptual fidelity of the generated images, as measured by perceptual oriented metrics that aim to capture human judgment of image quality. Moreover, this thesis proposes a simple end-to-end training framework that avoids domain transferability issues that arise when leveraging components pre-trained on natural video. Lastly, we show that the two main challenges for frame interpolation in this domain, large motion and information sparsity, interact such that the magnitude of objects' motion across frames conditions the appearance of artifacts associated with information sparsity.

Table of Contents

Page

Acknowledgements iv

Abstract v

Table of Contents vi

List of Tables ix

List of Figures x

Chapter

1 Introduction 1

2 Background and Related Work 3

2.1 Frame Interpolation 3

2.2 Optical Flow 4

2.3 Challenges of frame interpolation for 2D Animation 5

2.3.1 Large Nonlinear Motion 6

2.3.2 Information Sparsity 7

2.3.3 Spectral Bias/Texture Bias of neural networks 7

2.4 Proposed solutions to frame interpolation for 2D animation 9

2.4.1 Color Segmentation 9

2.4.2 Feature Extraction 9

2.4.3 Distance Transform 11

2.5 Non-correspondence vulnerability of optical flow 11

2.6 End-to-End Training for Sketches 13

2.7 UNet Autoencoder 14

2.8 Generative Adversarial Networks 15

2.8.1 Generative Adversarial Networks for Frame Interpolation 16

3 Proposed Approaches 18

3.1 System 1: UNet Interpolator . . . 18
3.1.1 Architecture . . . 18
3.1.2 UNet Loss Functions . . . 19
3.2 System 2: GAN Interpolator . . . 20
3.2.1 Discriminator Architecture . . . 21
3.2.2 Discriminator Loss Function . . . 22
3.2.3 Generator Loss Function . . . 22
4 Metrics & Dataset . . . 24
4.1 Reconstruction Metrics . . . 24
4.1.1 Peak Signal to Noise Ratio (PSNR) . . . 24
4.1.2 Structural Similarity Index Measure (SSIM) . . . 25
4.2 Perceptual Metrics . . . 26
4.2.1 Chamfer Distance . . . 26
4.3 Dataset . . . 27
4.3.1 ATD_12K dataset . . . 28
4.3.2 Sketch Extraction . . . 28
4.3.3 Augmentation . . . 29
5 Results . . . 30
5.1 Perceptual-Reconstruction Metric Trade-off . . . 30
5.2 Visualizing the effects of optimizing for perceptual metrics . . . 33
5.2.1 Visualizing generated frames across ranges of motion . . . 34
5.3 Training stability GAN vs UNet . . . 38
5.3.1 GAN Training . . . 38
5.3.2 UNet Training . . . 38
5.4 Balancing the perceptual-reconstruction trade-off . . . 38
5.5 Periodic activation functions to alleviate the Spectral Bias . . . 40
6 Concluding Remarks . . . 42
6.1 Significance of the Result . . . 42

6.2 Limitations 43
6.3 Future Work 43
References 45
Curriculum Vitae 52

List of Tables

5.1 Performance of both systems. Note that lower values are better for Chamfer Distance, and higher values are better for PSNR & SSIM. Chamfer Distance is scaled up by a factor of 10^4 for illustration purposes. 30

List of Figures

1.1 Example of key frames and in-between frames in an animated shot 1

2.1 A typical video triplet, with consecutive frames I_0, $I_{1/2}$, I_1 3

2.2 An example of optical flow vectors in an image. Each optical flow vector describes the trajectory that a point in an object has from one frame to the next [1] 4

2.3 A comparison of the results of interpolation using DAIN [2] for natural video (top) vs 2D animation (bottom). Severe artifacts can be observed for 2D animation. 5

2.4 An example of a triplet displaying small motion and a triplet displaying large motion. 6

2.5 An illustration of information sparsity, where the left shows an image with dense information, and the right shows an image with sparse information . 7

2.6 Illustration of the texture bias. When a neural network classifier is given conflicting shape and texture information in an image, it tends to resolve in favor of the texture information [13] 8

2.7 Figure 2.7 A diagram of Author Name's interpolation system 9

2.8 The interpolation system proposed by Chen at al. [7] 10

2.9 An example of Distance Transform 11

2.10 Mapping what pixels in the left column correspond to what pixels in the right column is a task much more suited for natural video than it is for 2D animation 12

2.11 Typical UNet Architecture [36] 14

2.12 Illustration of a GAN system [6] 16

3.1 UNet Interpolator architecture 19
3.2 Our GAN Interpolator System 21
3.3 Discriminator Architecture 22

4.1 Example of a triplet from the atd_12k dataset [movie: A Silent Voice] . . 28
4.2 Sketches of a triplet from the atd_12k dataset extracted using sketchKeras 28

5.1 Perceptual-reconstruction metric trade-off. As Chamfer Distance (left) becomes better, MSE (right) becomes worse 31
5.2 Chamfer Distance (lower is better) and SSIM (higher is better) are positively associated 32
5.3 Interpolation results of UNet (middle column) and GAN (rightmost column). It can be observed how the GAN system preserves details such as line structure and sharpness more to the likeness of the ground-truth 33
5.4 Small motion: GAN vs UNet 35
5.5 Medium motion: GAN vs UNet 36
5.6 Large motion: GAN vs UNet 37
5.7 GAN training instability: the discriminator loss (right) fails to converge for the training set (blue) and is extremely unstable in the validation set (orange) 39
5.8 UNet training stability: left shows training loss and right shows validation loss 39

Chapter 1

Introduction

Traditional 2D animation remains a largely manual process, where most of the frames in an animated sequence are hand-drawn. An average 23-minute animated TV episode will contain between 3,000 and 10,000 distinct hand-crafted drawings [8]. The pipeline to produce these drawings is usually divided into two stages: 1) drawing the key frames, which serve as the landmarks for a shot (usually the beginning, middle, and end of motion), and 2) drawing the in-between frames, which serve as intermediate frames that fill in the blanks from key frame to key frame. Professional animators often describe the latter process, a.k.a., in-betweening, as repetitive, monotonous, and creatively uninspiring work that is frequently relegated to inexperienced animators or outsourced to overseas production houses [24]. While the process of in-betweening has been largely automatized for other types of animation, such as 3D animation, there is not currently a robust solution to facilitate this process for traditional 2D animation.

Among the many challenges found in the automatization of in-betweening for 2D animation is the fact that professional 2D in-betweening is primarily done on sketches, that

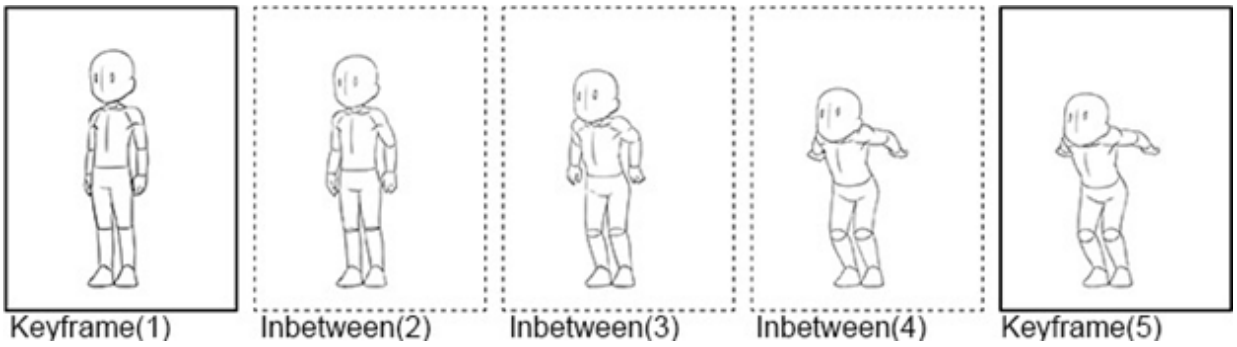

Figure 1.1: Example of key frames and in-between frames in an animated shot

is, uncolored drawings where the content of an image is conveyed only by lines [24]. Sketch is a unique domain with a host of challenges, mainly related to the sparsity of information (more discussion in 2.3.2). Furthermore, one of the preferred mediums for traditional 2D animators remains paper and pencil. This medium preference restricts potential solutions to the automatic in-betweening problem, as most of the research on sketches assumes the availability of vectorized drawings [51]. A vectorized drawing includes information on the direction and time of each stroke in the drawing, which can only be obtained if the drawing is created digitally.

Nonetheless, recent advances in computer vision, particularly those leveraging deep learning, a family of machine-learning approaches, have made strides in the automatization of related tasks within the same domain. Automatic colorization of sketches is not only a highly active area of research [37, 25, 45], but commercial implementations are now available to professional animators [5]. Moreover, deep-learning assisted rough sketch clean-up has been demonstrated to be feasible [39]. Furthermore, higher level tasks, such as character generation also show promising results [20]. Deep learning approaches acquired popularity relatively recently, therefore, most existing approaches to automatic 2D in-betweening are not learning-based and rely instead on hand-engineered rules [10] [43] [11] that often fail to generalize to diverse use cases. Moreover, the few deep-learning-based approaches for 2D in-betweening that have been published either require fully colored drawings [42] or expect vectorized input [19]. This study introduces a system that generates in-between frames for uncolored 2D animation using raster images (regular images), without the need for vectorized input.

Chapter 2

Background and Related Work

2.1 Frame Interpolation

The problem of automatic in-betweening is highly similar to the problem of frame interpolation, a widely studied problem in the computer vision community. Frame interpolation consists of generating intermediate frames in a video file, with the objective of increasing the video's frame rate (i.e., the number of images shown per second of video). The problem of frame interpolation can be formulated as finding a function S that takes two consecutive frames, I_n and I_{n+1}, as input and outputs a predicted intermediate frame $\hat{I}$. To train a frame interpolation system, the most common approach is to extract frame triplets, that is, sequences of 3 frames (I_0, $I_{1/2}$, I_1) from video files.

Figure 2.1: A typical video triplet, with consecutive frames I_0, $I_{1/2}$, I_1

In a typical frame interpolation system, the end frames of the triplet, I_0 and I_1 are used as input to make a prediction, $\hat{I}$, such that $S(I_0, I_1) = \hat{I}$. For learning-based systems,

during training, $\hat{I}$ is then compared to the real intermediate frame of the triplet, $I_{1/2}$, with the training objective being to minimize some error measure between $\hat{I}$ and $I_{1/2}$.

2.2 Optical Flow

One of the most prominent approaches to frame interpolation is the use of optical flow estimation [33]. Optical flow refers to the optical displacement of objects in a sequence of frames, which can also be thought of as the trajectory of the visual components in the frames.

Figure 2.2: An example of optical flow vectors in an image. Each optical flow vector describes the trajectory that a point in an object has from one frame to the next [1]

A typical optical flow estimation method will assess correspondences between pixels in consecutive frames, I_n and I_{n+1} and generate a map $f_{n\to(n+1)}$ of optical flow vectors that describe how each pixel in I_n needs to be displaced to reach its corresponding position in frame I_{n+1}. A standard flow-based frame interpolation method uses the estimated flow, $f_{n\to(n+1)}$, to guide a warping operation on image I_n that generates the predicted intermediate frame, $\hat{I}$. More sophisticated frame interpolation systems that use optical flow have a synthesis module that refines the initial prediction produced by the warping operation [18], [38]. Learning-based optical flow estimation methods are usually trained separately to the interpolation system by using annotated triplets with ground truth optical flow [9]. Nonetheless, there are also systems that extract the flow implicitly from non-annotated

triplets and learn in a self-supervised fashion from the frame interpolation error alone [17]. Self-supervised training refers to training done without the need for human annotation of the data. Given how expensive it is to obtain annotated flow data to train an optical flow estimation module, interpolation systems will often leverage pre-trained optical flow estimation networks [12, 42, 53].

2.3 Challenges of frame interpolation for 2D Animation

While state-of-the-art frame interpolation methods have achieved extraordinary results [35, 2, 16] for natural videos (recordings of real life), their success does not transfer well to the domain of 2D animation [30, 42]. There are several differences that can be pointed out between the 2D animation and the natural video domains. We outline the two most well documented challenges to performing frame interpolation for 2D animation, namely large motion and information sparsity.

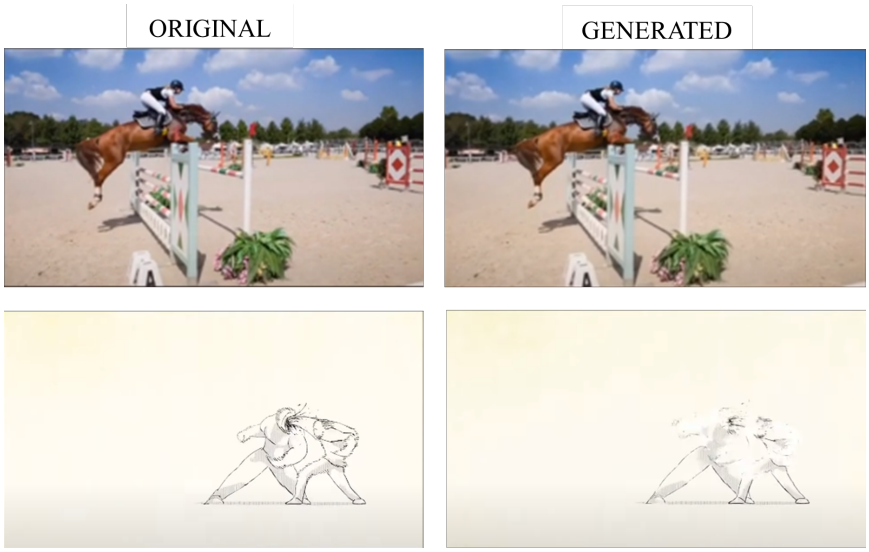

Figure 2.3: A comparison of the results of interpolation using DAIN [2] for natural video (top) vs 2D animation (bottom). Severe artifacts can be observed for 2D animation.

2.3.1 Large Nonlinear Motion

Large motion describes abrupt changes that can be observed when the objects being depicted show large displacement from one frame to the next. Large motion occurs substantially more in 2D animated videos than it does in natural videos. The reason being that in 2D animation the frames are hand-crafted drawings, which makes them significantly more expensive and time consuming to make than the photographic frames of a natural video. As a result, a standard natural video with a 24 frames per second rate (24 fps), will have 24 distinct frames in one second, while a standard 24 fps 2D animated video, will have between 6 to 12 distinct frames in one second. In other words, in 2D animation, each frame will be shown up to 4 times, with 2 being the norm for Japanese animation.

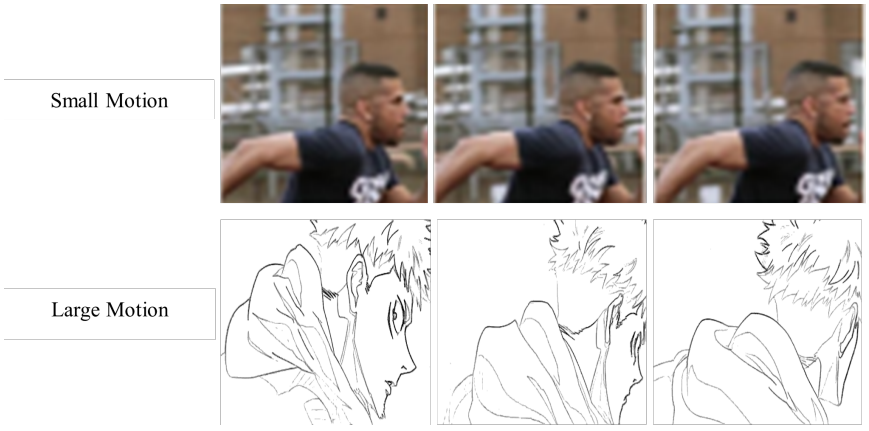

Figure 2.4: An example of a triplet displaying small motion and a triplet displaying large motion.

In addition, a key assumption that optical flow-based methods make is that the movement between frames in a triplet is linear. In the context of frame interpolation, linear movement means that objects in a video move in a straight line between two consecutive frames [33]. Consider pixel a in frame I_0 and flow vector given by $f_{0\to1}(a)$ that maps a's trajectory to its corresponding position in I_1. If a's corresponding position in intermediate frame, $I_{1/2}$, can be found by multiplying $f_{0\to1}(a)$ by some scalar, then the movement in that triplet is linear. Therefore, when performing flow-based interpolation, the warping

operation places each pixel in $\hat{I}$ at an intermediate point (usually the middle) along the pixel's displacement from I_n to I_{n+1}, given by $f_{n \to n+1}$. Due to the mechanical nature of video cameras, which shoot at very precise intervals the photographs that comprise a video, motion in natural video triplets can often be modeled linearly. Such is not the case for 2D animation, where the movement of objects is subject to animators' stylistic choices, which often result in non-linear motion.

2.3.2 Information Sparsity

The other main challenge for traditional 2D animation interpolation is the information sparsity inherent to uncolored drawings, or sketches [31]. In a natural image, every region of the image is packed with local fluctuations of information corresponding to color, textures, and other physical attributes of the photographed objects. In a sketch, on the other hand, the information is extremely sparse, as the image consists of mostly white space, with a minuscule portion of black pixels representing the lines that shape objects.

Figure 2.5: An illustration of information sparsity, where the left shows an image with dense information, and the right shows an image with sparse information

2.3.3 Spectral Bias/Texture Bias of neural networks

The challenge of information sparsity has particular relevance to deep-learning based approaches. It has been documented, both empirically and analytically, that neural networks favor the learning of one type of information over another. Empirically, it has been demonstrated that convolutional neural networks (CNNs) trained on natural images exhibit a

bias toward learning information encoded at spatially smaller scales (i.e., textures) over information encoded in larger scales (shapes and composition) [13]. In other words, CNNs trained on natural images exhibit a 'texture bias'. A proposed measure to alleviate this texture bias in a given CNN, is to train the network with 'shape cues' or images where texture information is not as abundant.

Figure 2.6: Illustration of the texture bias. When a neural network classifier is given conflicting shape and texture information in an image, it tends to resolve in favor of the texture information [13]

It was also demonstrated analytically, using tools from Fourier analysis, that networks with ReLU activations fit higher frequency functions significantly faster than lower frequency functions, thus exhibiting what was termed a "spectral bias" [34]. When transforming the visual content of an image to the frequency domain, visually abrupt changes from one region to the next, such as the edges of an object, yield high frequency values. Smoother, more gradual changes, such as the surface of an object, yield low frequency values. The sketch domain lacks any type of texture information, as the entirety of the information is conveyed by the edges that shape objects (i.e., high frequency changes). Therefore, when designing a frame interpolation system for 2D animation, leveraging networks pre-trained on natural images will entail combating a learned texture/spectral bias.

2.4 Proposed solutions to frame interpolation for 2D animation

2.4.1 Color Segmentation

In colored 2D animation, colored regions are often uniformly flat, lacking the local color fluctuations that give rise to the texture a natural image would have. Siyao et al. [42] decided to use it as an advantage by segmenting the frames into color regions, matching these regions, then using the matches to enrich the flow estimation. Color segmentation is a significantly easier task for colored 2D animated images than it would be for natural images. Their system then leverages a pre-trained flow estimation module that incorporates the color segment correspondences to refine the flow estimates.

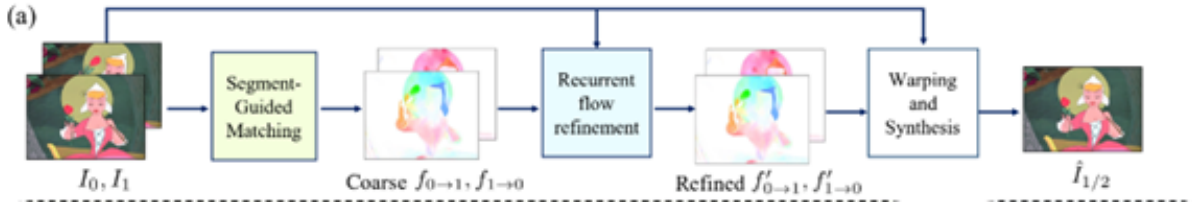

Figure 2.7: Figure 2.7 A diagram of Author Name's interpolation system

2.4.2 Feature Extraction

Chen et al. [7] continued this line of research and proposed an alternative system that exploits additional features of the colored 2D domain. In their system, Chen et al. use a pre-trained classifier, ResNet-50 [15], as a feature extractor, and a pre-trained flow estimation network, RAFT [44], to guide a warping operation. They then add a non-trainable inpainting module to handle occlusions (when an object blocks another object) and pass the results to a synthesis module (a UNet) that refines the predictions. Additionally, the authors leveraged the line information present in 2D animated drawings through a module

based on euclidean Distance Transform (DT), originally proposed by Narita et al. as a tool in sketch animation interpolation [30] (further discussion in 2.4.3). This distance transform module is executed concurrently, and its output is added to an additional synthesis module along with the predictions of the main module to further refine the final prediction.

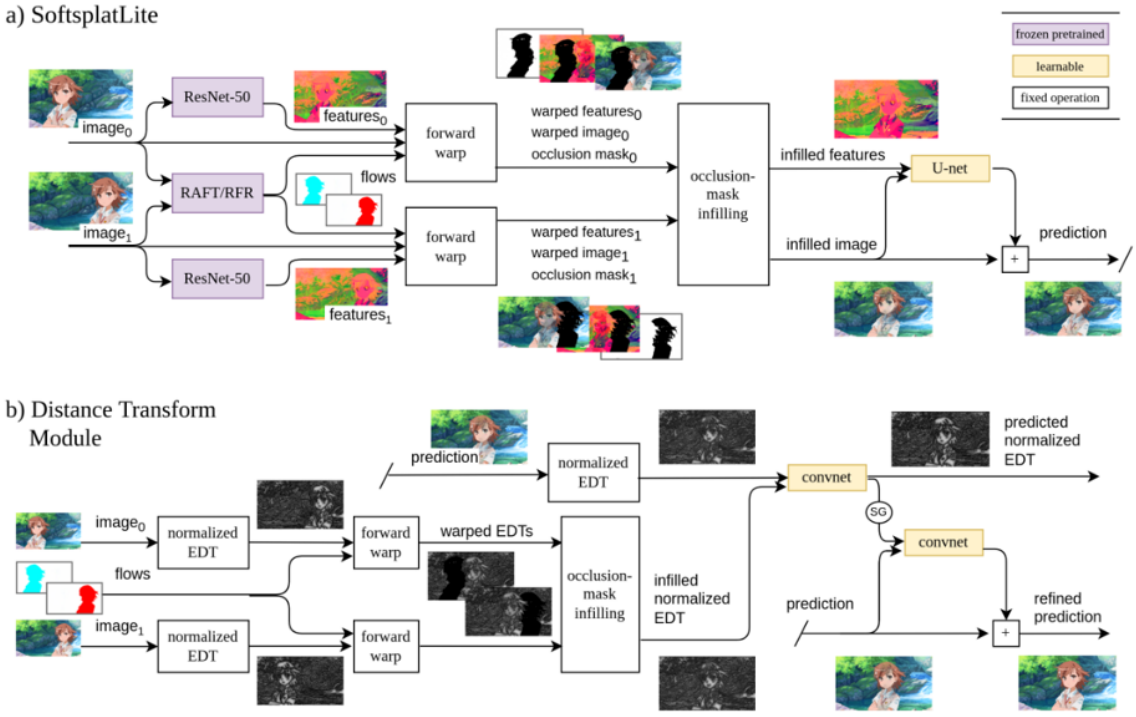

Figure 2.8: The interpolation system proposed by Chen at al. [7]

Approaches [42] and [7] achieve state-of-the-art results for colored 2D animation. They address large motion by enriching their flow estimation with either color correspondence information, or pre-trained feature matching. However, both approaches expect the images to be colored. Using colored frames allows them to leverage feature extraction and flow estimation models pre-trained on natural images, but makes them incompatible with the production pipeline of 2D animation, in which the animation is usually done on uncolored drawings.

2.4.3 Distance Transform

Narita et al. introduced an interpolation system specifically designed for uncolored 2D animated frames [30]. They alleviate the information sparsity issue by adding information to the frames via a Distance Transform (DT). DT replaces each white pixel in a binary image with the pixel's Euclidean distance to the closest black pixel. They then feed the transformed images to a pre-trained flow estimator. However, no training was performed in that system. Instead, the set of initial predictions made by the pre-trained optical flow estimator was evaluated directly.

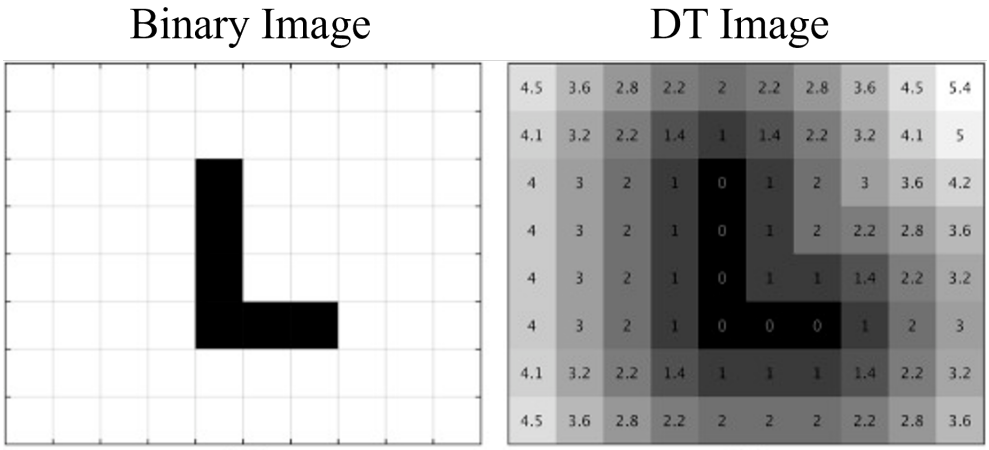

Figure 2.9: An example of Distance Transform

2.5 Non-correspondence vulnerability of optical flow

Large Motion and Information Sparsity are the two main issues documented in the literature of frame interpolation for 2D animation. The approaches discussed so far implement different strategies to address both. However, all these approaches use optical flow estimation, an approach that involves mapping pixel-wise correspondences across frames. We argue that, by nature, 2D animation frames will often lack the pixel-wise correspondences that natural video frames tend to have.

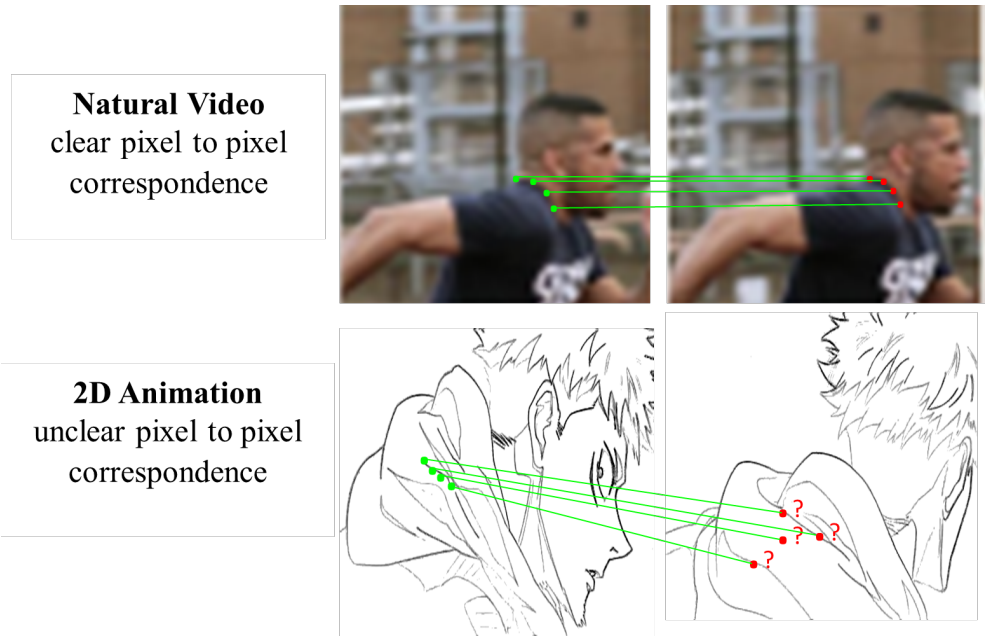

Figure 2.10: Mapping what pixels in the left column correspond to what pixels in the right column is a task much more suited for natural video than it is for 2D animation

Most properties in the objects of a natural video are strictly preserved from frame to frame. Animators draw each frame manually, and they use a much looser rendering criteria in which not all the properties of objects need to be preserved from frame to frame. For example, when drawing clothes, as long as the overall structure of the garment is conveyed, many other properties such as the exact number of folds and their shape need not be strictly preserved from one frame to the next, as observed in Figure 2.10.

Melvin et al. describe this type of frame to frame change common in 2D animation as a topological change [10]. They point out that when mapping corresponding objects across frames in 2D animation, it will be common for objects to be split into two or merged into one, to appear or disappear from one frame to the next; in other words, to undergo changes in their internal connectivity. They claim that a more strictly rendered object, such as one animated with a 3D modeling software, does not commonly undergo such changes from frame to frame.

We argue that this lack of consistency in objects' properties across frames makes pixel-

wise correspondence mapping, and by extension, optical flow, not the most well-suited approach to solve frame interpolation for 2D animation.

With flexibility in mind, we opt for end-to-end training, and implement two frame interpolation systems: 1) a stand-alone UNet-like autoencoder to be used as a baseline and 2) a framework based on Generative Adversarial Networks.

2.6 End-to-End Training for Sketches

End-to-end training, defined as the direct optimization of the entire system for a single task, has the property of requiring an amount of training data that grows proportionally with the complexity of the task. Thus, leveraging pre-trained models that have already been optimized for a related task is a widely used strategy to circumvent computational resources constraints. Nonetheless, for a domain as particular as sketch (uncolored) 2D animation, models pre-trained on other domains, such as the natural video domain, posit severe domain transferability challenges. These domain transferability challenges were discussed in depth in 2.3. Furthermore, we argue there is a property of the sketch domain that makes end-to-end training more feasible than it would be for the natural video domain. The information sparsity (discussed in 2.3.2) inherent to the sketch domain might raise spectral/texture bias issues, but it also produces a much smaller solution space. A natural RGB image has 3 channels per pixel, and each channel may have 255 different values. A binarized 2D sketch has only one channel, and 2 possible distinct values for that channel. Given that the solution space for uncolored or sketch 2D animated frames is substantially smaller than the solution space for natural video frames, we argue that end-to-end training is feasible, and worth exploring as an alternative to using texture/spectrally biased components pre-trained on natural images.

2.7 UNet Autoencoder

In an auto-encoder architecture, one half of the network encodes an image into a low-dimensional latent space, and the other half decodes this latent space into an image. UNet [36] is by far the most influential and widely used autoencoder architecture. A typical UNet has 4 contracting convolutional blocks that comprise the encoder, and 4 convolutional expanding blocks for the decoder. In order to preserve relevant information in the deeper layers of the network and avoid vanishing/exploding gradient issues common in deep neural networks, residual connections link blocks of corresponding scales.

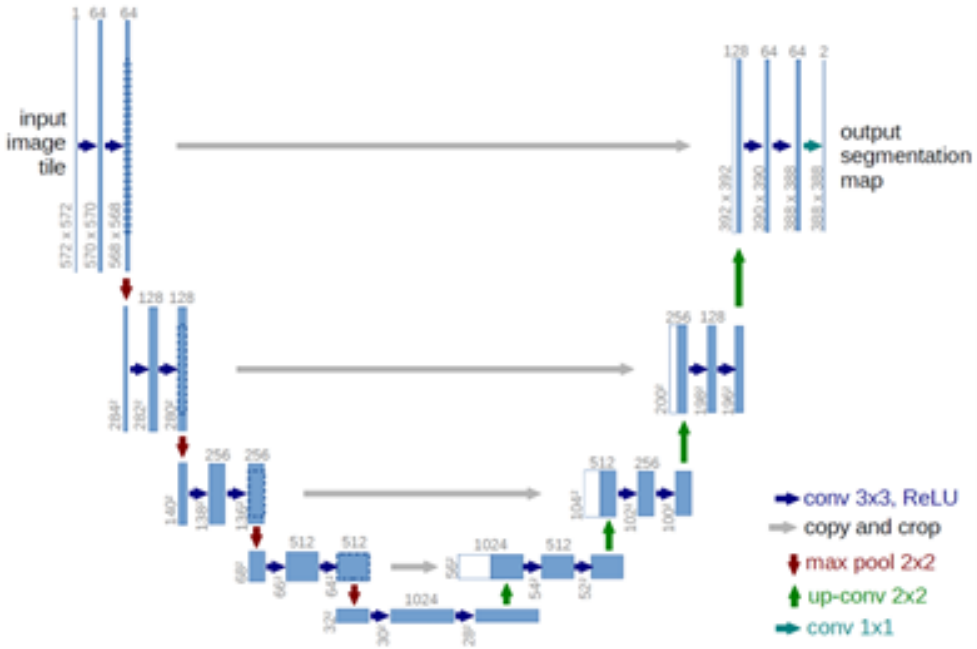

Figure 2.11: Typical UNet Architecture [36]

Residual connections have been shown to facilitate the learning of an identity function to retain un-changed properties that would otherwise be hard to preserve after applying a series of convolutional filters [15]. UNet was originally developed for medical image segmentation. However, it has since been used for a wide variety of applications, including frame interpolation [18, 35, 7]. In a frame interpolation scenario, UNet is usually just a

component of the system. For example, it is common for UNet to be used as a synthesis module, where a feature stack of optical flow features or other type of information gets passed as input, and the output is the final frame prediction. We show that for uncolored 2D animation, a standalone UNet, trained end-to-end without additional modules, can produce competitive frame interpolation results for small ranges of motion.

2.8 Generative Adversarial Networks

Generative Adversarial Networks (GANs) were first introduced in 2014 by Goodfellow et al. as a tool to generate realistic looking images from noise [14]. The central idea behind a GAN consists of having two neural networks, a generator G and a discriminator D, compete against each other. The generator is trained to produce fake images, and the discriminator is trained to distinguish the generator's fake images from real images. The losses of both generator and discriminator are setup in a zero-sum game, or minimax scenario, where minimizing the loss in the generator implies maximizing the error, or loss value of the discriminator, and vice-versa, the optimal state being a Nash equilibrium. As devised in the original paper [14], a GAN network optimizes the following function:

$$\min_G \max_D L(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.1}$$

Where x is a real image, and $D(x)$ is the discriminator network's prediction for that image, which ranges from 0 (fake) to 1 (real). $G(z)$ is a fake image produced by the generator given input z (in a typical GAN scenario z is a noise vector). Essentially, $(\mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)])$ measures, on average, how well the discriminator classifies real images as real, and $(\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$ measures how well it classifies fake images as fake. The discriminator network, D, seeks to maximize the overall value of the function. The generator network, G, on the other hand, seeks to minimize it. $D(G(z))$ is the discriminator's prediction for a fake image, and $1 - D(G(z))$ represents how well the discriminator

is fooled by the fake images. The generator wants $D(G(z))$ to be as close as possible to 1, as it would mean the discriminator is classifying its fake images as real. Therefore, the generator improves by producing more realistic images that can fool the discriminator into classifying them as real, and the discriminator improves by classifying the generator's fake images as fake and the real ones as real. Over the course of training, as both generator and discriminator improve at their respective tasks, the fake images produced by the generator start resembling the real images.

Several GAN-based architectures have been published since 2014 that can generate images nearly indistinguishable from real images to the human eye [47, 48, 49].

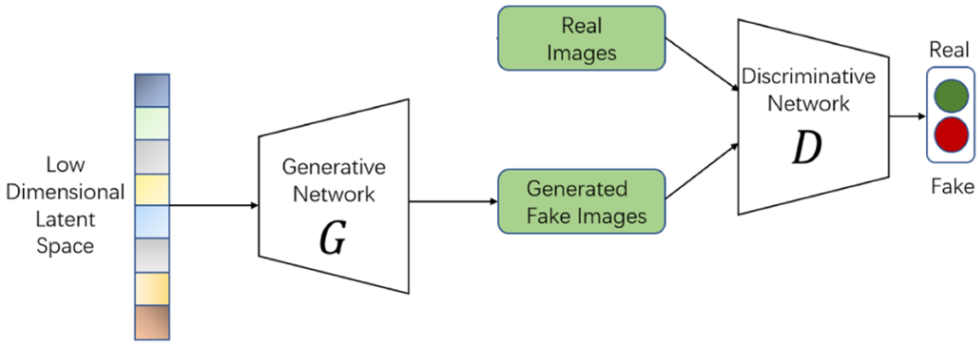

Figure 2.12: Illustration of a GAN system [6]

2.8.1 Generative Adversarial Networks for Frame Interpolation

The success of GANs did not take long to reach the field of frame interpolation. Several frame interpolation systems producing competitive results included a GAN-based component [49, 48, 47]. A common structure in GAN-based frame interpolation systems, is to use an auto-encoder architecture as the generator network, and a Convolutional Neural Network (CNN) as the discriminator [23, 46, 28, 48]. In an auto-encoder architecture, one half of the network encodes an image into a low-dimensional latent space, and the other half decodes this low-dimensional latent space into an image. In this type of GAN setup, the

auto-encoder receives the end frames of the triplet, I_0 and I_1, as input, and outputs a prediction, $\hat{I}$, for the in-between frame. The discriminator receives $\hat{I}$ and the real in-between frame, $I_{1/2}$, then provides an estimate of how realistic $\hat{I}$ is. In addition, supervised ground truth reconstruction error obtained by comparing $\hat{I}$ and $I_{1/2}$ directly is added to the generator's loss function. The supervised reconstruction error combined with the adversarial dynamic between the generator and the discriminator during training guides the generator toward producing plausible in-between frames. We argue that, if performed end-to-end, this approach to frame interpolation is less vulnerable to constraints such as the range and linearity of motion, as well as the lack of pixel-to-pixel correspondences of the training triplets.

Chapter 3

Proposed Approaches

We introduce 2 systems: 1) a simple UNet-like autoencoder, and 2) a GAN interpolator. Both systems are trained end-to-end, supervised by frame reconstruction error.

3.1 System 1: UNet Interpolator

3.1.1 Architecture

This system consists of a simple UNet-like autoencoder. We follow a typical UNet architecture with 4 contracting blocks for the encoder, and 4 expanding blocks for the decoder, and residual connections between blocks of corresponding scales. Each contracting block has 2 convolutional layers, followed by a max-pooling layer that halves the dimensions of the feature map for the next block. Our expanding blocks consist of a bilinear up-sampling transformation that doubles the size of the feature map and 3 convolutional layers. We use bilinear up sampling as opposed to transposed convolutions to avoid checkerboard artifacts, as suggested in [3]. Our autoencoder takes the end frames of a triplet I_0 and I_1, as input, and outputs a prediction for an in-between frame, $\hat{I}$. Reconstruction error is obtained by comparing $\hat{I}$ to ground-truth intermediate frame, $I_{1/2}$ .We use ReLU activations for all inner layers, and Sigmoid for the last layer.

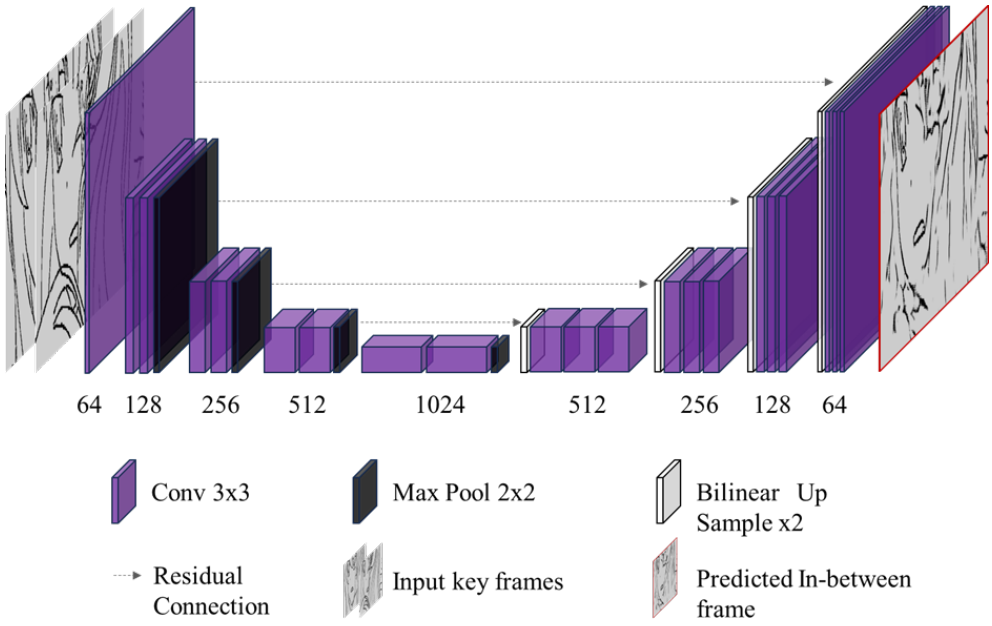

Figure 3.1: UNet Interpolator architecture

3.1.2 UNet Loss Functions

To encourage the restriction of the solution space to that of a binary image, we turn the reconstruction of each pixel into a classification problem, rather than a regression one. In other words, we are only interested in whether a pixel is fully dark or fully white, and would thus, need some measure of the binary accuracy of each pixel's classification. To this end, we optimize a Binary Cross Entropy (BCE) loss function, which is a differentiable proxy for binary accuracy, and average across all pixels of the reconstructed image. BCE Loss is defined as:

$$L_{\text{BCE}}(x, y) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(x_i) + (1 - y_i) \cdot \log(1 - x_i)] \tag{3.1}$$

Where N is the number of pixels in a binarized image, y_i is the ground truth label of the i^{th} pixel (0 for black and 1 for white) and x_i is the network's prediction for the i^{th} pixel's label (a value ranging between 0 and 1).

In addition to BCE, we use another loss function that looks into additional statistics of two images, termed Multi Scale Structural Similarity Index Metric (MS_SSIM), given by:

$$L_{MS\_SSIM}(x,y) = 1 - ([l_m(x,y)]^{\alpha} \prod_{j=1}^{m} [cs_j(x,y)]^{\beta_j}) \tag{3.2}$$

Where x and y are the two images being compared. $l_m(x,y)$ is the luminance component at the m^{th} scale. $cs_j(x,y)$ is the contrast-structure component function at the j^{th} scale. α and β_j are the weights for the luminance and contrast-structure comparisons, respectively.

The luminance component function is defined as:

$$l_m(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{3.3}$$

Where μ_x and μ_y are the means of x and y respectively. C_1 is a constant to avoid instability when the denominator is close to zero.

The contrast-structure component function is defined as:

$$cs_j(x,y) = \frac{2\sigma_{xy} + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{3.4}$$

Where σ_x^2 and σ_y^2 are the variances of x and y respectively. σ_{xy} is the covariance of x and y.

Both of these losses are integrated into an overall loss function in the following way:

$$L = \lambda_1 \cdot L_{\text{BCE}} + \lambda_2 \cdot L_{\text{MS\_SSIM}} \tag{3.5}$$

Where λ_1 and λ_2 are hyper-parameters that assign the weight each individual loss function has on the overall training objective.

3.2 System 2: GAN Interpolator

We follow a conventional GAN interpolator system with a UNet like autoencoder as the generator (see Figure 3.1) and a CNN as the discriminator (see Figure 3.2). The generator

combines a reconstruction loss by supervising its predictions with the ground-truth, and an adversarial loss from the discriminator. The discriminator works on a classification loss by comparing the generator's predicted in-between frames against the ground-truth in-between frames. In essence, the generator optimizes 2 objectives: 1) fooling the discriminator, and 2) reconstructing the ground truth in-between frames. The discriminator in turn optimizes the regular classification objective between the generator's fake frames and the ground truth frames.

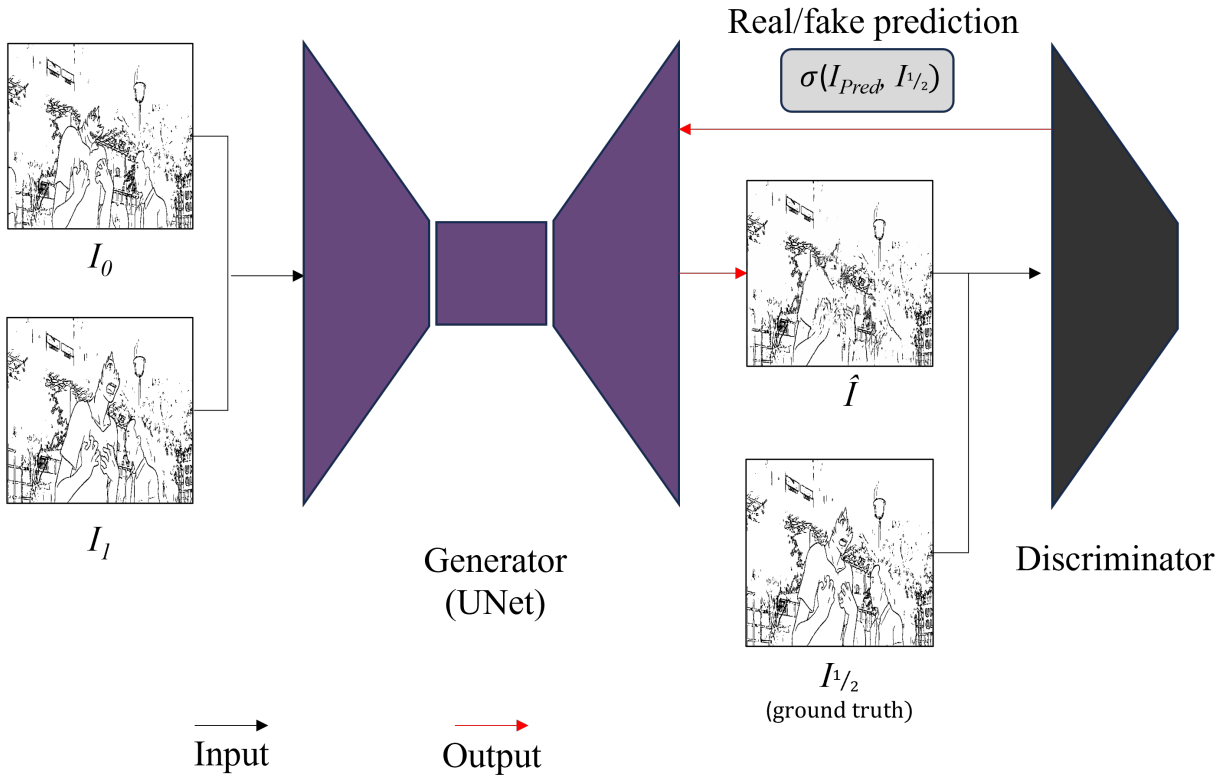

Figure 3.2: Our GAN Interpolator System

3.2.1 Discriminator Architecture

For the discriminator, we follow the typical classifier CNN structure of halving the dimensions of the feature map at every convolutional block, then aggregating the features of the last convolutional layer with a dense layer before classification. To avoid over-

parametrization, we keep the discriminator network light by using only a single convolutional layer per block.

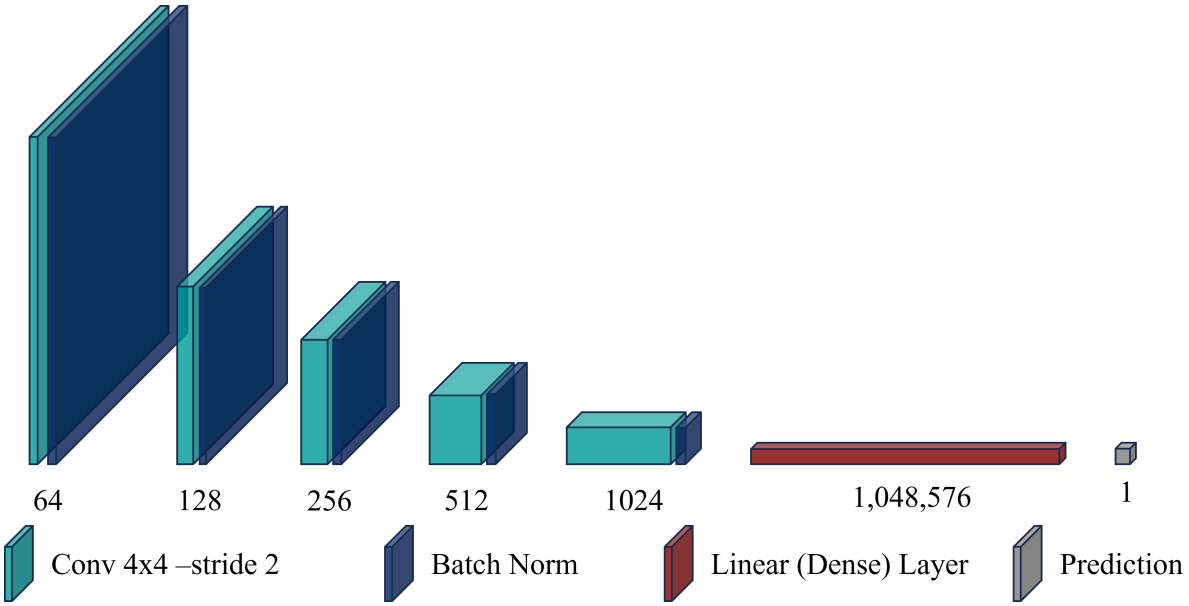

Figure 3.3: Discriminator Architecture

3.2.2 Discriminator Loss Function

The discriminator optimizes a Binary Cross Entropy (BCE) as defined in 3.1.2. However, in this case, the classification is done over entire images, instead of over individual pixels, and the labels are 0 for a fake image and 1 for a real image.

3.2.3 Generator Loss Function

The generator network is the part of the system that produces the frame. Our generator network's loss is divided into two parts optimizing the following objectives: 1) frame reconstruction loss, obtained through supervision from the ground truth intermediate frames, and 2) adversarial loss, obtained through the discriminator network's classification results.

1. For our reconstruction objective, just like with the UNet system, we use L_{BCE} (see 3.1.2) and L_{MS_SSIM} (see 3.1.2).

2. For our adversarial objective, L_{avd}, we use the discriminator's classification results, obtained after passing both the generator's fake images, and the real intermediate frames $\hat{I}$ to the discriminator, as formulated in the original paper, (see 2.8), which simplifies to the following expression for the generator:

$$L_{\text{adv}}(\hat{I}) = -\frac{1}{N}\sum_{i=1}^{N}\log(D(\hat{I}_i)) \tag{3.6}$$

The generator network in this GAN system integrates all sub-losses with the following loss function:

$$L_G = \lambda_1 \cdot L_{\text{BCE}} + \lambda_2 \cdot L_{\text{MS\_SSIM}} + \lambda_3 \cdot L_{\text{adv}} \tag{3.7}$$

Where λ_1, λ_2, and λ_3 are hyper-parameters that assign the weight each individual loss function has on the overall training objective. We found through experimentation that low values of λ_3 ($>.05$), that is, the weight assigned to the adversarial loss, L_{avd}, lead to more accurate error-based reconstruction metrics, while higher values lead to better perceptual-based metrics (more discussion in 4). Nonetheless, this parameter is highly sensitive, as increments as low as 5% in λ_3 can significantly interfere with model convergence.

Chapter 4

Metrics & Dataset

Frame interpolation research has mostly been evaluated with image reconstruction metrics, however, several of these measures have been shown to not necessarily align with human perception of image quality. Evaluation of frame interpolation systems remains an active research area, with new metrics being proposed continuously.

4.1 Reconstruction Metrics

Classical reconstruction metrics used by the frame interpolation community include Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM).

4.1.1 Peak Signal to Noise Ratio (PSNR)

PSNR is essentially a normalized version of Mean Squared Error (MSE), which is just the squared of the error obtained after subtracting the pixel intensity values in one image from the pixel intensity values in the other image and averaging the result. MSE between two images is given by:

$$\text{MSE}(I, \hat{I}) = \frac{1}{n} \sum_{i=1}^{n} (I_i - \hat{I}_i)^2 \tag{4.1}$$

Where I_i is the i^{th} pixel intensity of the ground truth image, $I_{1/2}$, that has n pixels, and $\hat{I}_i$ is the i^{th} pixel intensity of the predicted image $\hat{I}$ with the same number of pixels. The MSE of a testing dataset is just the average of the individual MSEs.

MSE is a very straightforward metric, however, the range of values it produces is not

standardized, thus, interpreting them is highly subject to the domain and dataset on which the metric is being used (natural images, spectral images, 3d renders, etc.). PSNR re-scales MSE according to the images' highest possible pixel value, performs a logarithmic transformation and a simple algebraic manipulation that transform MSE from a distance metric, where lower values are better, into a peak signal (highest possible pixel value) to noise (error as measured by MSE) ratio, where higher values are better. PSNR is given by:

$$\text{PSNR} = 20 \cdot \log_{10} \left(\frac{\text{MAX}_I}{\sqrt{\text{MSE}}} \right) \tag{4.2}$$

Where MAX_I is the highest possible pixel value for the given dataset, which in our case is a pixel intensity of 1, since we normalize our images from the 0 to 255 pixel intensity range, to the 0 to 1 range.

4.1.2 Structural Similarity Index Measure (SSIM)

SSIM is a more involved metric that combines several statistics of 2 images, x and y, into a single quantity in the following way:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{4.3}$$

Where μ is an image's mean pixel intensity value, σ is the standard deviation, and c_1, c_2, and c_3 are just constants. The formula is derived from multiplying the following 3 components by each other: luminance $*$ contrast $*$ structure.

Luminance is defined as:

$$L_m(x, y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \tag{4.4}$$

Contrast is defined as:

$$\text{ct}(x, y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \tag{4.5}$$

Structure is defined as:

$$\text{st}(x, y) = \frac{2\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \tag{4.6}$$

It can be observed that the luminance component, $L_m(x, y)$, penalizes the metric when the mean pixel values of the images being compared are different from each other. Contrast, $ct(x, y)$, applies the same mechanism, but for the images' standard deviations. Structure, $st(x, y)$ is plainly the correlation between all pixels in both images.

One of the sublosses employed by both our systems, L_{MS_SSIM} (see 3.1.2) is a derivation of this metric, which adds multi-scale comparisons and combines some of these components with each other. Nonetheless, as an evaluation metric, SSIM is more commonly used than MS_SSIM.

4.2 Perceptual Metrics

While impressive interpolation results have been achieved with classical reconstruction metrics like PSNR and SSIM, it has been recently shown that optimizing for these metrics does not cleanly align with human judgments of perceptual quality [4]. Thus, the evaluation of interpolation quality remains an open research problem. A widely adopted perceptual based metric in frame interpolation and GAN research is the Learned Percceptual Image Patch Similarity (LPIPS) [54]. LPIPS consists of passing the images being compared through a pre-trained classifier network, such as VGG-19 [40] to obtain a representation vector from the network's convolutional feature maps, then computing the distance between the representation vectors of the two images. While LPIPS has shown remarkable correlates with human judgment of image reconstruction quality [54], most pre-trained classifier networks being used to compute this metric were trained with natural images. We outlined our reasons to avoid leveraging components pre-trained on natural images in section 2.3.

4.2.1 Chamfer Distance

Chamfer distance is a metric typically used to measure similarity between point clouds and is popular in fields such as 3D rendering [32] [52]. For images, this metric uses objects' edges to perform something akin to shape matching. Narita et al. proposed the use of this

metric for the evaluation of frame interpolation for uncolored 2D animation [30]. Chen at al. also used it as one of their metrics in their state-of-the-art interpolation system for colored 2D animation [7].

The Chamfer Distance between two sets, X and Y, is given by:

$$\text{Chamfer Distance}(X, Y) = \frac{1}{2|X|} \sum_{x \in X} \min_{y \in Y} |x - y| + \frac{1}{2|Y|} \sum_{y \in Y} \min_{x \in X} |y - x| \tag{4.7}$$

To adapt it to binary images, Narita et al use a Distance Transform (DT) on the images, which replaces every white pixel in a binary drawing with the pixel's distance to the closest black pixel. See 2.4.3 for an illustration of DT. Using the same formulation as [30], we perform DT on our predicted frame, $\hat{I}$, and our ground truth frame, $I_{1/2}$, and obtain the Chamfer Distance (CD) in the following way:

$$\text{CD}(\hat{I}, I_{1/2}) = \frac{1}{2} \left(\frac{I_{1/2} \cdot \text{DT}(\hat{I})}{|I_{1/2}|} + \frac{\hat{I} \cdot \text{DT}(I_{1/2})}{|\hat{I}|} \right) \tag{4.8}$$

The DT is a non-differentiable operation, hence, it is not straight-forward to optimize for this objective directly with a loss function. Chen at al. proposed performing DT for both $\hat{I}$, and $I_{1/2}$ without tracking the gradients of this operation, then optimizing the Laplacian Pyramid loss between the DT of both frames [7]. We tested this loss, termed L_{DT}, however, we found via experimentation that it had minimal impact in any of our systems' Chamfer Distance.

4.3 Dataset

A positive aspect of the frame interpolation problem is that obtaining datasets to perform supervised training is relatively easy, as triplets with ground truth in-between frames can be extracted from practically any video. There are several tools such as shot-detection algorithms that help to decompose a video into its constituent triplets.

4.3.1 ATD_12K dataset

The authors' of [42] assembled and annotated a dataset of 12,000 2D animation triplets, namely, the 'atd_12k' dataset. The triplets were collected from a series of Western and Japanese 2D animated movies. The criteria they used to collect this dataset was to consider any set of three consecutive frames with a SSIM value within the range [0.65, 0.95] as a triplet. This dataset has since been used as a benchmark for 2D animation interpolation research [7] [26]. We train and test our model with the atd_12k dataset, splitting it into 9,000 triplets for training, 1,000 for validation, and 2,000 for testing.

Figure 4.1: Example of a triplet from the atd_12k dataset [movie: A Silent Voice]

4.3.2 Sketch Extraction

The atd_12k is a colored animation dataset, and our system is meant to perform interpolation on uncolored 2D animation. To this end, we extract sketches, or uncolored frames, from the atd_12k dataset by using sketchKeras [27], a model trained to extract the line art from colored 2D illustrations.

Figure 4.2: Sketches of a triplet from the atd_12k dataset extracted using sketchKeras

4.3.3 Augmentation

Data augmentation is the process of creating new training sample samples by applying some transformation(s) to the original samples with the goal of increasing the diversity and robustness of the dataset. For our dataset, we perform two simple augmentations: horizontal flipping (inverting the image over the vertical axis), and reversing the order of the triplet (from I_0, $I_{1/2}$, I_1 to I_1, $I_{1/2}$, I_0). The augmentation scheme is applied randomly such that 50% of the training batches undergo any one or both of these transformations.

Chapter 5

Results

System	Chamfer Distance ▼	PSNR ▲	SSIM ▲
UNet - full	1.673	14.048	0.789
GAN - full	**0.984**	**15.081**	**0.852**

Table 5.1: Performance of both systems. Note that lower values are better for Chamfer Distance, and higher values are better for PSNR & SSIM. Chamfer Distance is scaled up by a factor of 10^4 for illustration purposes.

We tested 2 systems: 1) a standalone UNet interpolator and 2) GAN based interpolator, which adds an adversarial component to the UNet interpolator. Results show that adding this adversarial component significantly improves the model performance both in terms of perceptual and reconstruction-based metrics. Nonetheless, the performance improvement is substantially more noticeable for the perceptual metric (42% better Chamfer Distance).

5.1 Perceptual-Reconstruction Metric Trade-off

In this project, a clear distinction was made between between perceptual-based evaluation, which are designed to align more with human judgments of image quality (see 4.2), and reconstruction-based evaluation, which more directly assess pixel-level errors in the generation (see 4.1) We found during training in our GAN system, that as performance on perceptual metrics increased, performance on reconstruction metrics decreased. This reconstruction-perceptual metric trade-off has been documented in the literature [54]. Theoretically, GANs are designed to produce more 'realistic' results to the human eye, which

aligns with what perceptual metrics capture, but these perceptual metrics are often at odds with classical reconstruction metrics. Results of our experiments fall in line with this documented trade-off. Figure 5.1 shows how, in our GAN system, as the perceptual metric, Chamfer Distance (CD), becomes better during training (lower is better for CD), the reconstruction metric, MSE becomes worse (PSNR is derivative of MSE).

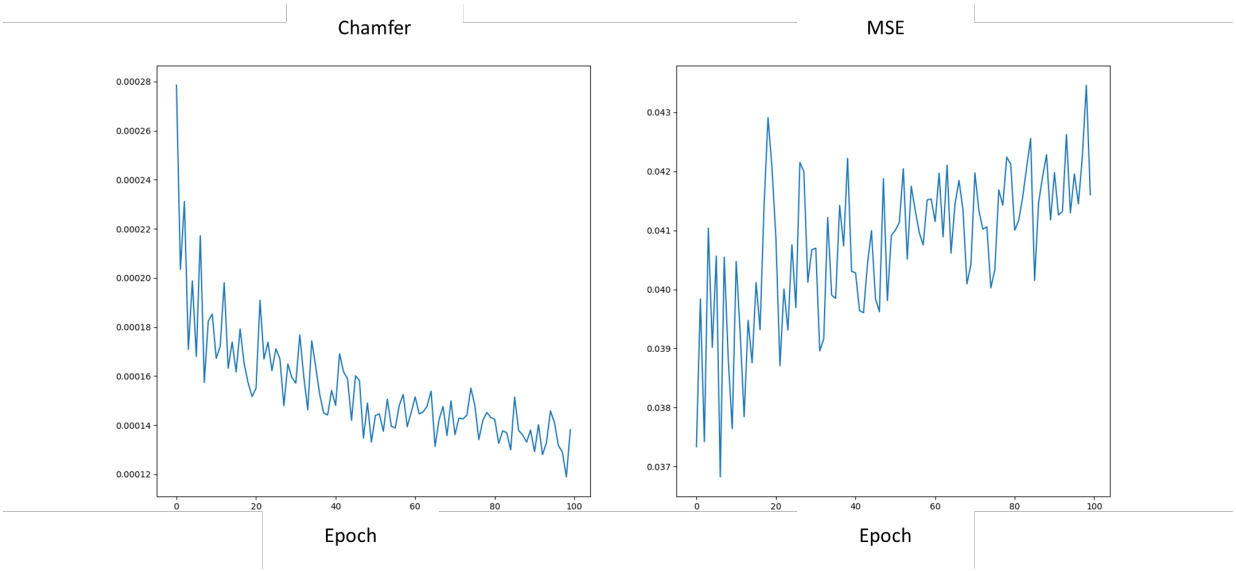

Figure 5.1: Perceptual-reconstruction metric trade-off. As Chamfer Distance (left) becomes better, MSE (right) becomes worse

This trade-off, however, did not apply to both reconstruction metrics utilized in this study. We found SSIM and Chamfer Distance (CD) to have a positive association across our experiments. We observed that lower (better) CD values were associated with higher (better) SSIM values across multiple experiments, both with the UNet and GAN systems. In contrast to PSNR, which essentially measures normalized pixel-to-pixel error, SSIM combines various image statistics into a single metric, which possibly makes it more robust to comparisons with perceptually oriented metrics like CD.

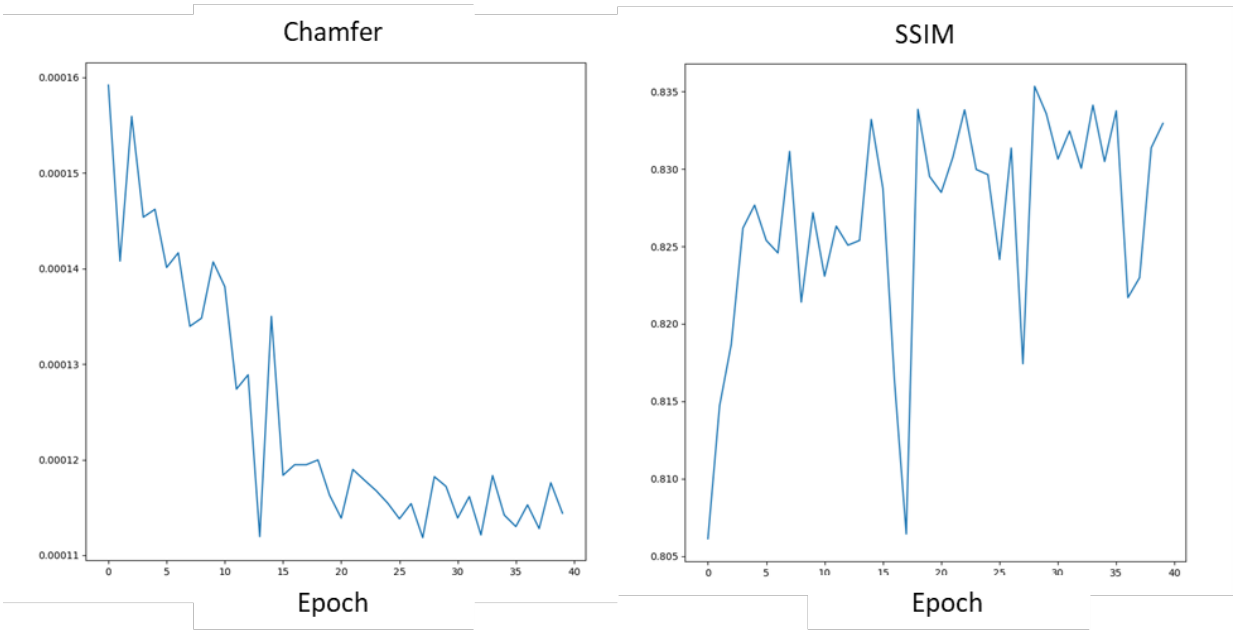

Figure 5.2: Chamfer Distance (lower is better) and SSIM (higher is better) are positively associated

5.2 Visualizing the effects of optimizing for perceptual metrics

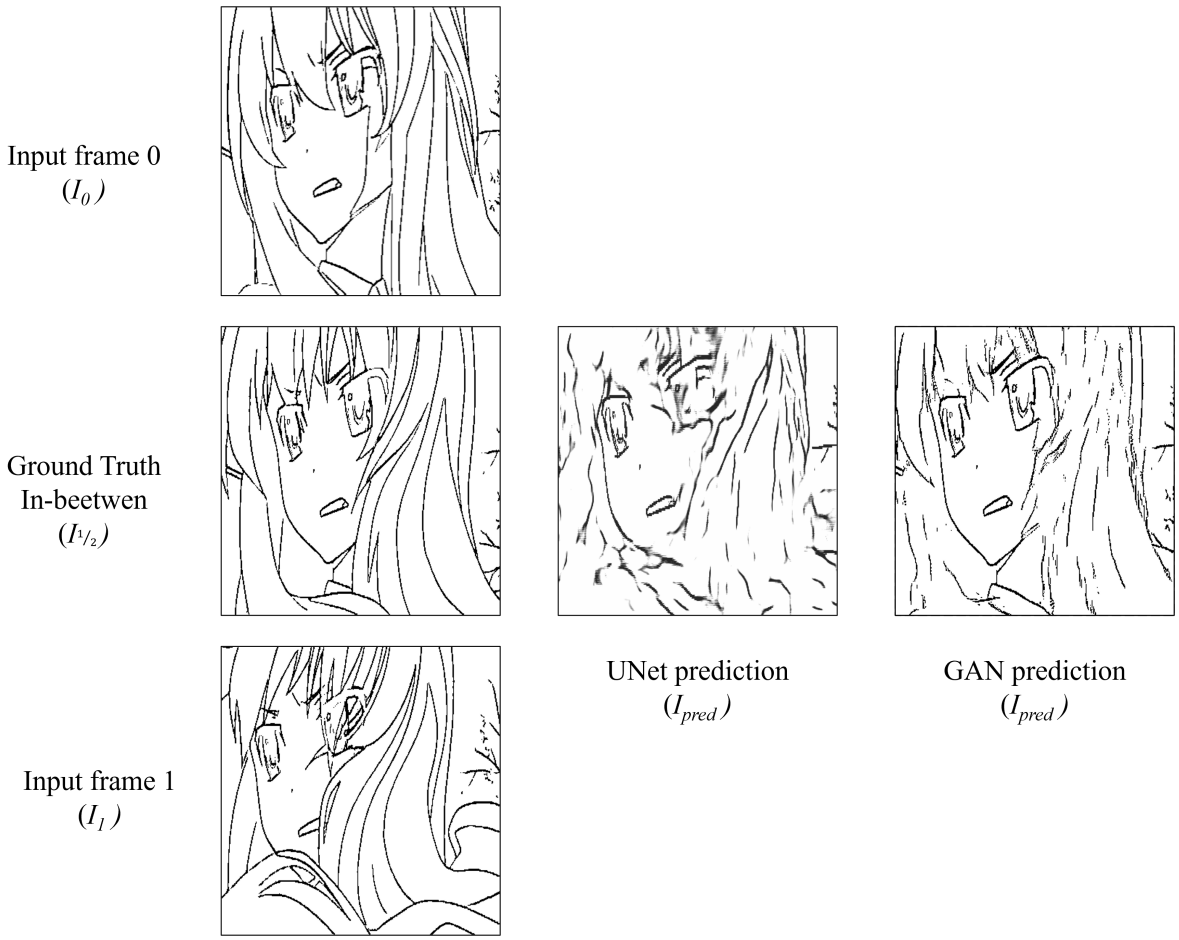

Figure 5.3: Interpolation results of UNet (middle column) and GAN (rightmost column). It can be observed how the GAN system preserves details such as line structure and sharpness more to the likeness of the ground-truth

As can be observed in Figure 5.3, for large motion triplets, the GAN interpolator preserves certain aspects of the original drawing, such as line thickness, sharpness, and overall geometry that more closely resemble the original drawings. The UNet interpolator on the other hand, is prone to blurring artifacts and deformations to line thickness and curvature.

We argue these differences can be attributed to the distinct training objectives of both systems: UNet's loss function consists of purely of error-based measurements between its generation and the ground truth. A GAN's objective, on top of that same error based objective, includes the adversarial component where a discriminator network dynamically looks for the presence/absence of learned features that distinguish generated frames from real frames.

5.2.1 Visualizing generated frames across ranges of motion

While the GAN-based system shows better overall testing performance over the UNet-based system, this difference in performance is most observable to the human eye only past some ranges of motion. Movement range, discussed in 2.3.1, is roughly defined as the displacement length of objects across frames. When the movement in a triplet is small, predicted in-between frames from both the UNet and the GAN interpolators do not display clearly noticeable differences (see Figure 5.4). However, as the movement range increases, there are clearly noticeable differences between the GAN reconstruction and the UNet reconstruction (see Figure 5.5). We prepared a miniature example dataset entirely for visualization purposes, comprised of 1 small motion triplet, 1 middle motion triplet, and 1 large motion triplet. We visualize the performance of the light models compared to that of the full models, by showing each model's reconstructions of these examples.

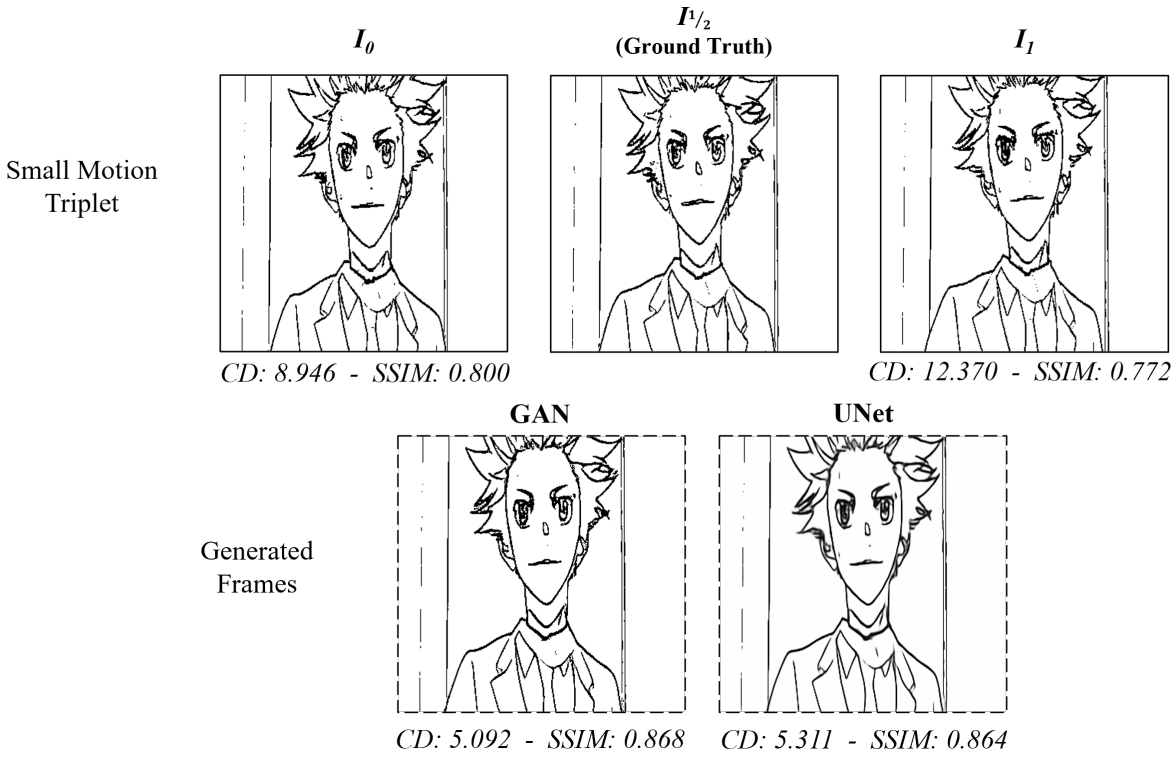

Figure 5.4: Small motion: GAN vs UNet

As can be observed in Figure 5.4, for triplets with small motion, both models produce generations that are virtually indistinguishable from each other to the naked eye. The similarity of both generations is also reflected in their metric values, which are closely aligned with each other.

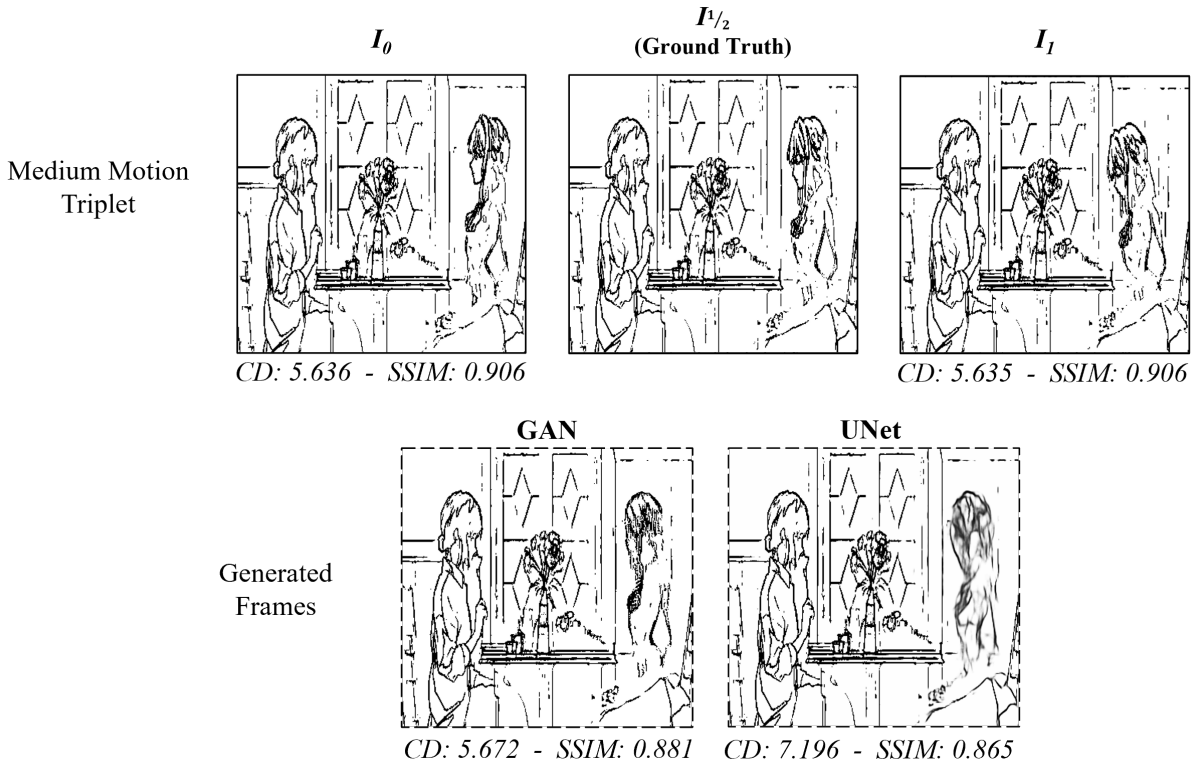

Figure 5.5: Medium motion: GAN vs UNet

For triplets with medium range of motion (see Figure 5.5), the differences between the generated frames by the GAN and the UNet models become much more evident. The GAN model produces crisper lines and an apparent geometry that's more consistent with the ground truth. The UNet model seems to default to blurring the lines, possibly as a result of its training objective much more oriented toward minimizing the number of incorrectly classified white pixels.

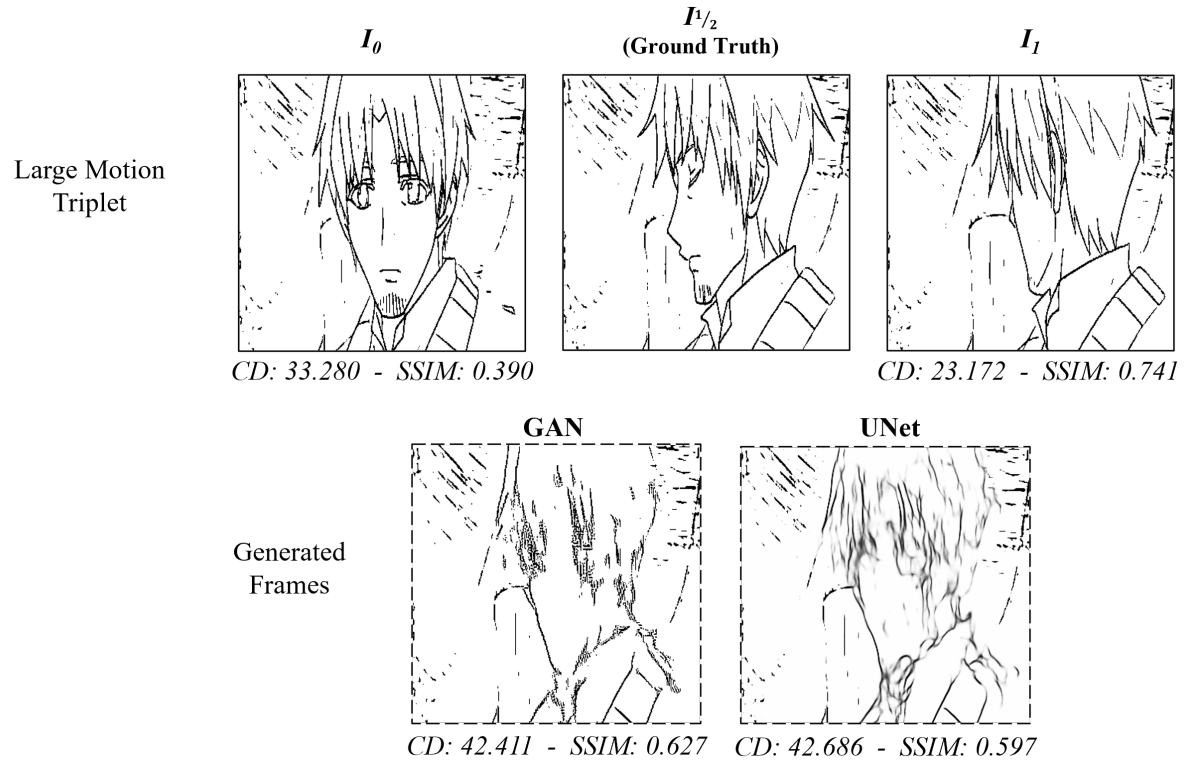

Figure 5.6: Large motion: GAN vs UNet

For triplets where the range of motion is large, both approaches struggle to produce convincing results (see Figure 5.6), while displaying different classes of artifacts. The GAN system produces notably more disappearing artifacts, that is, entire regions incorrectly left blank. The UNet system, on the other hand, produces more blurred regions, with geometrically deformed lines. A potential explanation for this difference in artifacts is that the discriminator network in the GAN system may learn to penalize the generator for producing lines without the proper sharpness of curvature more heavily than it penalizes it for an over-abundance of white space, seeing as a the ground truth frames are, on average, 97% white space. The UNet system, on the other hand, optimizes an exclusively error-based objective, in which it tries to match both pixel-to-pixel values, as well as certain statistics of both images. Thus, blurring entire regions is possibly an effective strategy to minimize the pixel-to-pixel error, while maintaining some image statistics, like the mean

pixel value, relatively consistent.

5.3 Training stability GAN vs UNet

5.3.1 GAN Training

One caveat of GANs is their notable lack of training stability, particularly of the discriminator network (see Figure 5.7). Several strategies have been proposed to stabilize the discriminator network during GAN training, one of them being Spectral Normalization [29], which essentially re-scales the weights of an entire layer with the largest singular value of the layer's weight matrix. We use spectral normalization in all layers of our generator and discriminator and find notable benefits in training stability. However, the stability was highly conditioned to careful selection of hyper-parameters and loss functions. We found that small variations in hyper-parameters, such as the lambda parameters corresponding to the weights assigned to each sub-loss quickly lead to highly unstable training, even when using spectral normalization.

5.3.2 UNet Training

Unlike GAN systems, simple auto-encoders, such as the UNet variations we used, enjoy much more stable training. We found UNet training to be substantially less involved, as most of our experiments reached convergence with much more stability (see Figure 5.8).

5.4 Balancing the perceptual-reconstruction trade-off

In the GAN-based system, the weight of the adversarial loss, L_{adv}, on the overall training objective (see 3.2.3), is explicitly determined by the hyperparameter λ_3. We found through experimentation that increases to λ_3 produce higher perceptual metric values at the expense of reconstruction metrics, while low λ_3 values (<0.05) facilitate convergence at lower recon-

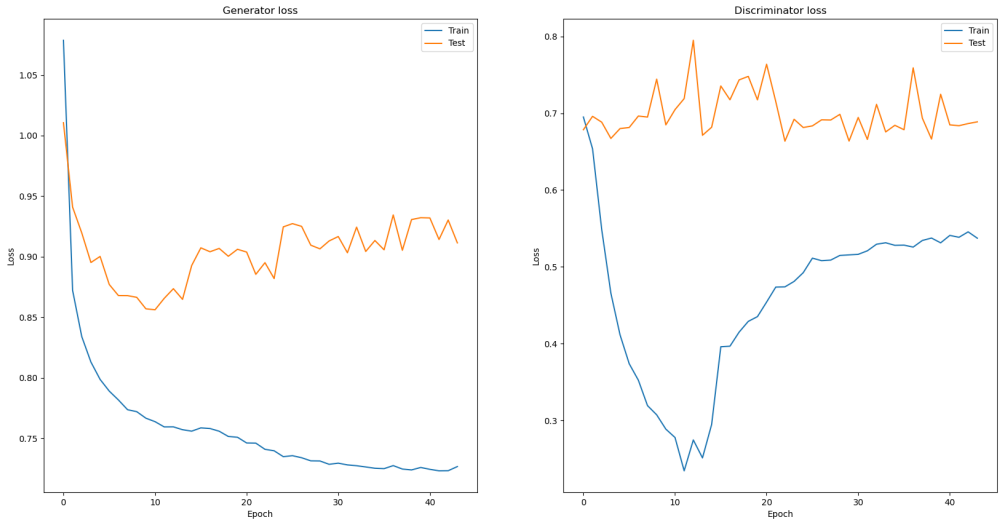

Figure 5.7: GAN training instability: the discriminator loss (right) fails to converge for the training set (blue) and is extremely unstable in the validation set (orange)

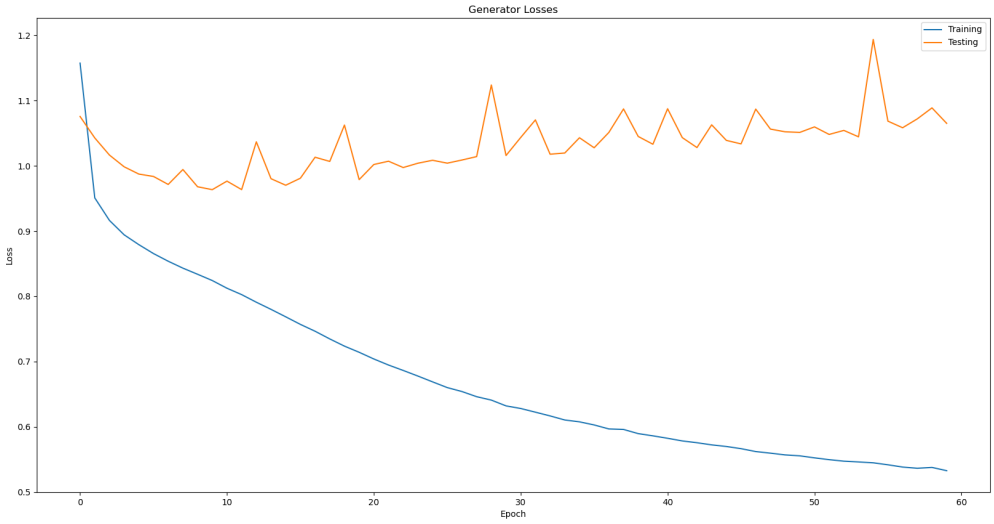

Figure 5.8: UNet training stability: left shows training loss and right shows validation loss

struction metrics, at the expense of perceptual based metrics, effectively serving as a tool to regulate the perceptual-reconstruction trade-off (see 5.1). Nonetheless, in line with the literature documenting the instability of GAN training [29], the λ_3 parameter is extremely sensitive, as fluctuations of as little as 5% can interfere with training loss convergence.

5.5 Periodic activation functions to alleviate the Spectral Bias

An explicit measure developed to alleviate the Spectral Bias is the use of sine activation functions, introduced by Sitzman et al. in their implicit neural representation networks, termed Siren [41]. The authors report that the Siren networks they introduce are able to fit images and functions in such a way that the first and second order derivatives of the data are accurately fitted by the implicit neural representation of their network. In the case of images, information concerning the edges of objects is conveyed strongly through the derivative of the data's representation (pixel value matrix) with respect to their spatial coordinates (x, y). We used the sine activations reported in this paper, along with their corresponding initialization scheme, as indicated by the authors. Nonetheless, while the use of sine activations sped up the fitting of the training data for small datasets, it made our models extremely vulnerable to mode collapse, thus failing to converge to competitive metrics for most of our experiments. Mode collapse is a term used to describe the model converging to fit one mode of the data distribution. For example, in our experiments, when we used sine activations, the models defaulted to reconstruct one of the 2 input frames it received, as opposed to the intermediate frame it is supposed to generate. These results might be indicative of the need for a more carefully designed strategy to integrate sine activations with Convolutional Neural Networks (CNNs). When introduced by Sitzman et al., sine activations were used with a notably different type of network: implicit neural representations, which are essentially networks that take an image's coordinates as input,

as opposed to its pixel values, and do not typically employ convolutions during image generation.

Chapter 6

Concluding Remarks

This study demonstrates the feasibility of using Generative Adversarial Networks, as well as autoencoders in an end-to-end fashion to generate in-between frames for 2D animation, particularly for triplets with small to middle motion. It was observed that adding an adversarial component to a UNet autoencoder siginificantly improves its performance, particularly in terms perceptual metrics aimed at capturing human perceptions of image quality.

Furthermore, we show that this perceptual improvement of the GAN system is most observable in triplets with medium motion magnitude across frames. For triplets with small ranges of motion, both approaches, the UNet-based system and the GAN-based system, produce equally convincing results. For large motion, however, the GAN system produces artifacts typically associated with information sparsity, such as disappearing objects. These findings suggest that the two main challenges for frame interpolation in this domain, large motion and information sparsity, interact in such a way that the negative effects of information sparsity are exacerbated by increases in motion magnitude.

Therefore, we argue that range of motion is the main challenge to overcome. Future research directions for this problem would benefit from modeling longer range dependencies of the feature maps.

6.1 Significance of the Result

This study introduces the first deep learning system, to the authors' knowledge, that performs in-between frame generation on uncolored 2D animation from raster images that can

be trained end-to-end. Furthermore, we show that adding an adversarial component to an autoenconder architecture produces substantial improvements in the perceptual quality of the interpolated results, while also improving reconstruction quality.

6.2 Limitations

The proposed methods in this study were not compared to other methods designed to perform interpolation on uncolored 2D animation. This is partially due to computational resources constraints, as several of these methods are comprised of substantially larger networks than the ones we used. Additionally, the benchmark dataset used, atd_12k [42] was manually filtered to contain only frame triplets with linear motion (see 2.3.1 for a definition and discussion of motion linearity). The proposed systems were designed with versatility in mind, therefore, future studies should assemble a dataset unfiltered for motion linearity to asses the systems' ability to model non-linear motion. Additionally, future studies should also compare these systems to other established frame interpolation systems.

6.3 Future Work

To further alleviate the spectral bias neural networks are vulnerable to, also known as texture bias, which the uncolored 2D animation domain is particularly vulnerable to, a promising approach is the addition of positional encodings. This can be achieved through the use of Fourier features that have been successfully applied in Transformer models and implicit neural representations [50]. Similarly, it may be beneficial to experiment with sine activations within the context of implicit neural representations, by replacing our convolutional decoder with a Siren network [41]. This could potentially enhance the model's ability to deal with the sparse information inherent to uncolored 2D animation.

To tackle the challenge of large motion, a multi-scale training apprach could be em-

ployed. This is a a standard approach used in state-of-the-art frame interpolation [35] [49] as well as GAN research [21] [22] as it helps the model better understand and represent large-scale movements in the frames. The downsampling operation required to perform multi-scale training loses a lot of information when performed on uncolored 2D animated frames, given the sparsity of the information in these frames. However, Distance Transform could be used to increment the information present in the frames and make them more robust to down sampling, thus allowing for multi-scale training. Alternatively, the inclusion of self-attention layers could assist with large motion handling. Self-attention mechanisms have proven effective in various domains for capturing long-range dependencies and could be particularly useful in this context.

In terms of data, collecting a larger dataset could significantly improve the model's performance and robustness. Findings in this study echo findings from prior studies [7] in that naively collecting triplets from animated video hurts the models' performance. Therefore, a classifier network could be designed and trained to assist in filtering a larger dataset collected from animated videos.

References

[1] Abhijit. An introduction to the nvidia optical flow sdk. `https://developer.nvidia.com/blog/an-introduction-to-the-nvidia-optical-flow-sdk/`, 2019.

[2] 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.

[3] Sara Björk, Jonas Nordhaug Myhre, and Thomas Haugland Johansen. Simpler is better: spectral regularization and up-sampling techniques for variational autoencoders. *arXiv preprint arXiv:2201.07544*, 2022.

[4] Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6228–6237, 2018.

[5] CadmiumCD. Cadmium app, 2023. Accessed: Oct 16, 2023.

[6] Likun Cai, Yanjie Chen, Ning Cai, Wei Cheng, and Hao Wang. Utilizing amari-alpha divergence to stabilize the training of generative adversarial networks. *Entropy*, 22(4):410, 2020.

[7] Shuhong Chen and Matthias Zwicker. Improving the perceptual quality of 2d animation interpolation. In *European Conference on Computer Vision*, pages 271–287. Springer, 2022.

[8] C. Coats. How is anime made?, 2023. Accessed: May 20, 2023.

[9] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet:

Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[10] Melvin Even, Pierre Bénard, and Pascal Barla. Non-linear rough 2d animation using transient embeddings. In *Computer Graphics Forum*, volume 42, pages 411–425. Wiley Online Library, 2023.

[11] Jean-Daniel Fekete, Érick Bizouarn, Éric Cournarie, Thierry Galas, and Frédéric Taillefer. Tictactoon: A paperless system for professional 2d animation. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 79–90, 1995.

[12] Pedro Figueirêdo, Avinash Paliwal, and Nima Khademi Kalantari. Frame interpolation for dynamic scenes with implicit flow encoding. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 218–228, 2023.

[13] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Real-time intermediate flow estimation for video frame interpolation. In *European Conference on Computer Vision*, pages 624–642. Springer, 2022.

[17] Junhwa Hur and Stefan Roth. Optical flow estimation in the deep learning age. *Modelling Human Motion: From Human Perception to Robot Design*, pages 119–140, 2020.

[18] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9000–9008, 2018.

[19] Jie Jiang, Hock Soon Seah, and Hong Ze Liew. Stroke-based drawing and inbetweening with boundary strokes. In *Computer Graphics Forum*, volume 41, pages 257–269. Wiley Online Library, 2022.

[20] Yanghua Jin, Jiakai Zhang, Minjun Li, Yingtao Tian, Huachun Zhu, and Zhihao Fang. Towards the automatic anime characters creation with generative adversarial networks. *arXiv preprint arXiv:1708.05509*, 2017.

[21] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863, 2021.

[22] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[23] Mark Koren, Kunal Menda, and Apoorva Sharma. Frame interpolation using generative adversarial networks, 2017.

[24] kViN. Anime's present and future at stake: The in-betweener problem, 2020. Accessed: Oct 16, 2023.

[25] Yeongseop Lee and Seongjin Lee. Automatic colorization of anime style illustrations using a two-stage generator. *Applied Sciences*, 10(23):8699, 2020.

[26] Yu-Lun Liu, Yi-Tung Liao, Yen-Yu Lin, and Yung-Yu Chuang. Deep video frame interpolation using cyclic frame generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8794–8802, 2019.

[27] lllyasviel. Sketchkeras, 2017. Accessed: 2023-02-01.

[28] Rishik Mishra, Neeraj Gupta, and Nitya Shukla. Fregan: an application of generative adversarial networks in enhancing the frame rate of videos. *arXiv preprint arXiv:2111.01105*, 2021.

[29] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[30] Rei Narita, Keigo Hirakawa, and Kiyoharu Aizawa. Optical flow based line drawing frame interpolation using distance transform to support inbetweenings. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4200–4204. IEEE, 2019.

[31] Rei Narita, Koki Tsubota, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using deep features. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 3, pages 49–53. IEEE, 2017.

[32] Trung Nguyen, Quang-Hieu Pham, Tam Le, Tung Pham, Nhat Ho, and Binh-Son Hua. Point-set distances for learning representations of 3d point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10478–10487, 2021.

[33] Anil Singh Parihar, Disha Varshney, Kshitija Pandya, and Ashray Aggarwal. A comprehensive survey on video frame interpolation techniques. *The Visual Computer*, pages 1–25, 2022.

[34] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

[35] Fitsum Reda, Janne Kontkanen, Eric Tabellion, Deqing Sun, Caroline Pantofaru, and Brian Curless. Film: Frame interpolation for large motion. In *European Conference on Computer Vision*, pages 250–266. Springer, 2022.

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[37] Min Shi, Jia-Qi Zhang, Shu-Yu Chen, Lin Gao, Y Lai, and Fang-Lue Zhang. Reference-based deep line art video colorization. *IEEE Trans. Vis. Comput. Graph*, 20(1), 2022.

[38] Joi Shimizu, Heming Sun, and Jiro Katto. Forward and backward warping for optical flow-based frame interpolation. In *2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 082–086. IEEE, 2022.

[39] Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. Learning to simplify: fully convolutional networks for rough sketch cleanup. *ACM Transactions on Graphics (TOG)*, 35(4):1–11, 2016.

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[41] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

[42] 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.

[43] Daniel Sỳkora, John Dingliana, and Steven Collins. As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proceedings of the 7th International Symposium on Non-photorealistic Animation and Rendering*, pages 25–33, 2009.

[44] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020.

[45] Harrish Thasarathan, Kamyar Nazeri, and Mehran Ebrahimi. Automatic temporally coherent video colorization. In *2019 16th conference on computer and robot vision (CRV)*, pages 189–194. IEEE, 2019.

[46] Quang Nhat Tran and Shih-Hsuan Yang. Efficient video frame interpolation using generative adversarial networks. *Applied Sciences*, 10(18):6245, 2020.

[47] 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.

[48] Shiping Wen, Weiwei Liu, Yin Yang, Tingwen Huang, and Zhigang Zeng. Generating realistic videos from keyframes with concatenated gans. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(8):2337–2348, 2018.

[49] Jian Xiao and Xiaojun Bi. Multi-scale attention generative adversarial networks for video frame interpolation. *IEEE Access*, 8:94842–94851, 2020.

[50] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural

fields in visual computing and beyond. In *Computer Graphics Forum*, volume 41, pages 641–676. Wiley Online Library, 2022.

[51] Peng Xu, Timothy M Hospedales, Qiyue Yin, Yi-Zhe Song, Tao Xiang, and Liang Wang. Deep learning for free-hand sketch: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):285–312, 2022.

[52] Rongfei Zeng, Mai Su, Ruiyun Yu, and Xingwei Wang. Cd2: Fine-grained 3d mesh reconstruction with twice chamfer distance. *ACM Transactions on Multimedia Computing, Communications and Applications*, 19(6):1–25, 2023.

[53] Dacheng Zhang, Weimin Lei, Wei Zhang, and Xinyi Chen. Flow-based frame interpolation networks combined with occlusion-aware mask estimation. *IET Image Processing*, 14(17):4579–4587, 2020.

[54] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

Curriculum Vitae

Francisco Arriaga Pazos earned his Bachelor's degree on psychology, with a minor in mathematics on 2019, from the University of Texas at El Paso. After a year working as a statistical research analyst for El Paso Community College, he decided to pursue graduate studies and enrolled in the Master's of Science in Data and Information Sciences program at the University of Texas at El Paso. His research interests lie on the intersection of the artistic and the technical domains, which spans topics such as content-based recommender systems, and generative modeling for digital artwork. Under the guidance of Dr. Olac Fuentes, he studied frame interpolation systems for 2D animation, while working as a statistical research analyst for El Paso Community College.

email: fran.arp@hotmail.com