Enforcing Realism and Temporal Consistency for Large-Scale Video Inpainting

by

Ryan B. Szeto

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2021

Doctoral Committee:

Professor Jason J. Corso, Co-Chair
Associate Professor Honglak Lee, Co-Chair
Assistant Professor Justin Johnson
Assistant Professor Andrew Owens

Ryan B. Szeto

szetor@umich.edu

ORCID iD: 0000-0002-2966-7138

DEDICATION

This dissertation would not exist without the help of those who have fundamentally shaped my educational career. First, I would like to thank Daniel Gabriner, Dennis McCowan, and Prof. Jack Wileden for introducing me to computer science and for teaching me the joy of logical problem solving. Next, I want to thank Profs. Rick Adrion, Timothy Richards, and Paul Dickson for fostering both my primary undergraduate research and my interest in graduate school. In addition, I wish to thank the friends made during my undergraduate and graduate careers, as well as my therapist Jessica Berner, for supporting me during the lows and highs of my education. Furthermore, I would like to thank my mother and father for teaching me the importance of education, dedication, and respect for others. Last but not least, I want to thank the squirrels of U-M for teaching me the beauty of nature and for helping me retain my mental health during unprecedented life challenges.

ACKNOWLEDGEMENTS

Thank you to the committee members—Prof. Jason J. Corso, Prof. Honglak Lee, Prof. Justin Johnson, and Prof. Andrew Owens—for providing advice and constructive feedback throughout the dissertation process.

The bi-TAI project was partly supported by ARO W911NF-15-1-0354, DARPA FA8750-17-2-0112, and DARPA FA8750-16-C-0168. It reflects the opinions and conclusions of its authors, but not necessarily the funding agents.

The HyperCon project was partially completed while the author was a research scientist intern at Samsung Semiconductor, Inc. The Moving Symbols project was partially completed while the author was a research scientist intern at Toyota Research Institute.

The DEVIL project greatly benefited from the exploratory contributions made by Xuetong Sun and Jing-An Tzeng. The material is based upon work supported by the National Science Foundation under Grant No. 1628987. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

Chapter

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Today, people are consuming more videos than ever before. At the same time, video manipulation has rapidly been gaining traction due to the influence of viral videos, as well as the convenience of editing software. Although video manipulation has legitimate entertainment purposes, it can also be incredibly destructive. In order to understand the positive and negative consequences of media manipulation—as well as to maintain the integrity of mass media—it is important to investigate the capabilities of video manipulation techniques.

In this dissertation, we focus on the manipulation task of video inpainting, where the goal is to automatically fill in missing parts of a masked video with semantically relevant content. Inpainting results should possess high visual quality with respect to reconstruction performance, realism, and temporal consistency, i.e., they should faithfully recreate missing contents in a way that resembles the real world and exhibits minimal flickering artifacts.

Two major challenges have impeded progress toward improving visual quality: *semantic ambiguity* and *diagnostic evaluation*. Semantic ambiguity exists for any masked video due to several plausible explanations of the events in the observed scene; however, prior methods have struggled with ambiguity due to their limited temporal contexts. As for diagnostic evaluation, prior work has overemphasized aggregate analysis on large datasets and underemphasized fine-grained analysis on modern inpainting failure modes; as a result, the expected behaviors of models under specific scenarios have remained poorly understood.

Our work improves on both models *and* evaluation techniques for video inpainting, thereby providing deeper insight into how an inpainting model's design impacts the visual quality of its outputs. To advance state-of-the-art in video inpainting, we propose two novel solutions that improve visual quality by expanding the available temporal context. Our first approach, *bi-TAI*, intelligently integrates information from multiple frames before and after the desired sequence. It produces more realistic results than prior work, which could only consume limited contextual information. Our second approach, *HyperCon*, suppresses flickering artifacts from frame-wise processing by identifying and propagating consistencies found in high frame-rate space; we successfully apply it to tasks as disparate as video inpainting and style transfer.

Aside from methodological improvements, we also propose two novel evaluation tools to diagnose failure modes of modern video inpainting methods. Our first such contribution is the

Moving Symbols dataset, which we use to characterize the sensitivity of a state-of-the-art video prediction model to controllable appearance and motion parameters. Our second contribution is the *DEVIL benchmark*, which provides a dataset and a comprehensive evaluation scheme to quantify how several semantic properties of the input video and mask affect video inpainting quality.

Through models that exploit temporal context—as well as evaluation paradigms that reveal fine-grained failure modes of modern inpainting methods at scale—our contributions enforce better visual quality for video inpainting on a larger scale than prior work. We enable the production of more convincing manipulated videos for data processing and social media needs; we also establish replicable fine-grained analysis techniques to cultivate future progress in the field.

CHAPTER 1

Introduction

1.1 Background

If a picture is worth a thousand words, then videos are worth orders of magnitude more. Since they engage both our visual and aural senses across time, videos can be incredibly expressive and emotionally provocative. Nowadays, they form a major part of our daily lives. Consider the last time you watched a video to entertain yourself, learn a new skill, or understand an ongoing news story; it was likely within the past week, if not the past 24 hours. They are so important that at least 50% of people under the age of 30 say that they "do not know how they would get through life without video" [1].

People are creating and consuming more videos now than ever before [2], which we believe is largely due to recent trends in social media and content creation software. Social media platforms such as YouTube and TikTok have made it incredibly easy to share videos with massive audiences. At the same time, video content creation is now quick and convenient due to modern editing apps, which allow creators to modify videos by splicing clips, overlaying text, applying low-level filters, and more. During their inception, video editors were rare, complex, and limited to desktop computers [3]; however, they are now incredibly common (Figure 1.1), easy to use [4], and widely

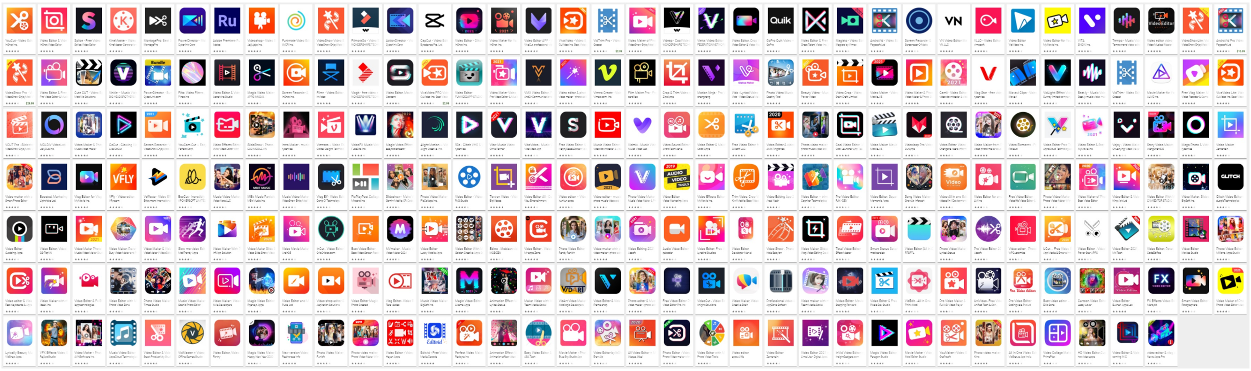


Figure 1.1: Search results for "video editor" on Google Play. Hundreds of easy-to-use video editors are available in phone app stores.

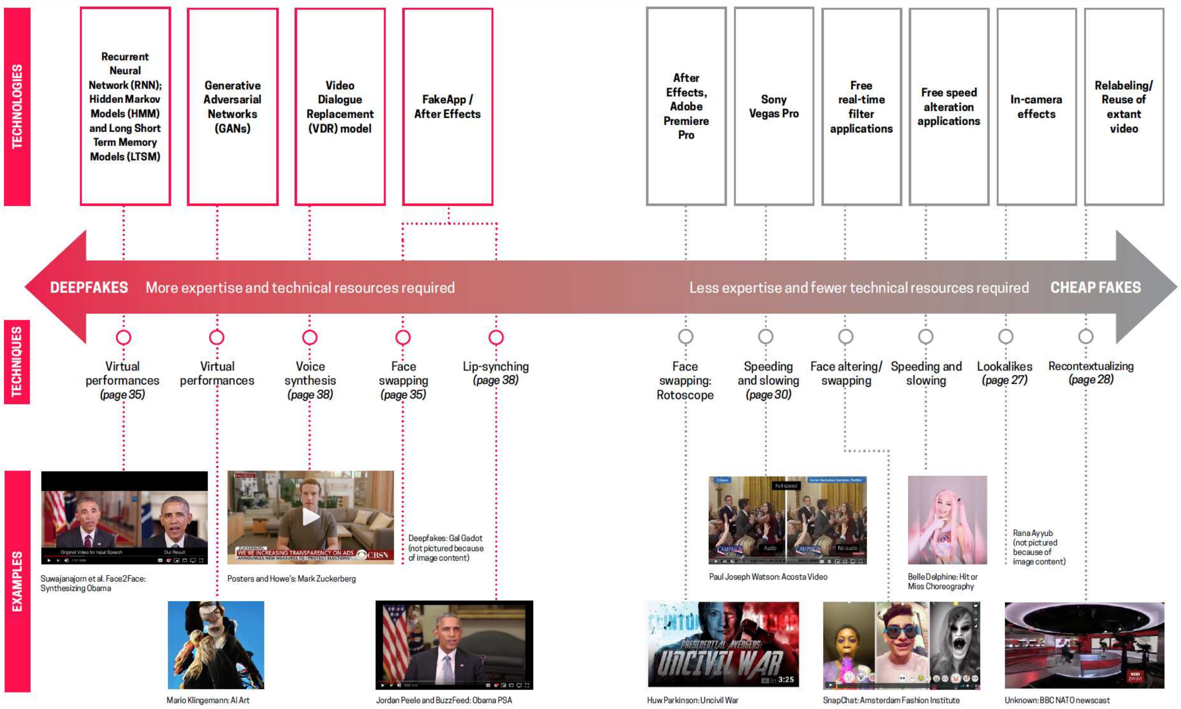

TECHNOLOGIES
Recurrent Neural Network (RNN); Hidden Markov Models (HMM) and Long Short Term Memory Models (LTSM)
Generative Adversarial Networks (GANs)
Video Dialogue Replacement (VDR) model
FakeApp / After Effects
After Effects, Adobe Premiere Pro
Sony Vegas Pro
Free real-time filter applications
Free speed alteration applications
In-camera effects
Relabeling/ Reuse of extant video
DEEPFAKES
More expertise and technical resources required
Less expertise and fewer technical resources required
CHEAP FAKES
TECHNIQUES
Virtual performances (page 35)
Virtual performances
Voice synthesis (page 38)
Face swapping (page 35)
Lip-synching (page 38)
Face swapping: Rotoscope
Speeding and slowing (page 30)
Face altering/ swapping
Speeding and slowing
Lookalikes (page 27)
Recontextualizing (page 28)
EXAMPLES
Suwajanajorn et al. Face2Face: Synthesizing Obama
Posters and Howe's: Mark Zuckerberg
Deepfakes: Gal Gadot (not pictured because of image content)
Mario Klingemann: AI Art
Jordan Peele and BuzzFeed: Obama PSA
Paul Joseph Watson: Acosta Video
Belle Delphine: Hit or Miss Choreography
Rana Ayyub (not pictured because of image content)
Huw Parkinson: Uncivil War
SnapChat: Amsterdam Fashion Institute
Unknown: BBC NATO newscast

Figure 1.2: Video manipulation techniques exist across a spectrum of required expertise and resources. In this diagram, techniques decrease in complexity from left to right. Figure modified from Paris and Donovan [12]; both the source and the adapted figure are available under the CC BY-NC-SA 4.0 license.

available on phones. The popularity and accessibility of social media and video editing software have helped cement videos as a significant portion of our current media consumption.

Given the influence of video and the accessibility of video editors, it is no wonder that video manipulation has gained substantial attention in creative content and national media [5]–[8]. Video manipulation refers to the broad set of techniques that alter videos to achieve specific outcomes. They exist across a wide range of complexity (Figure 1.2) and are remarkably popular. For instance, Instagram "filters"—whose abilities range from applying low-level filters to making subjects appear more attractive with cosmetic changes (despite heavy criticism regarding mental health [9])—have helped the app grow its user base to over 1 billion monthly active users [10]. Even single-purpose manipulation apps, e.g., for artistic stylization, have achieved remarkable success, claiming more than 100 million installations [11].

Video manipulation based on deep neural networks (DNNs) has become especially prevalent in the public eye due to the alarmingly convincing nature of manipulated content. They are often used in humorous situations, e.g., to put Nicolas Cage in popular movies [13] or to reveal Tom Cruise's passion for industrial cleaning [5], [14]. However, they can also be used to harm victims, e.g., by creating pornographic content [6], [7] or by humiliating political figures [8], [15]. The severity of these negative applications has mobilized organizations in academia, industry, and the government to counteract them [16]–[18]. Understanding the full capabilities of video manipulation techniques is crucial to developing countermeasures that will help protect society and the integrity of mass media.

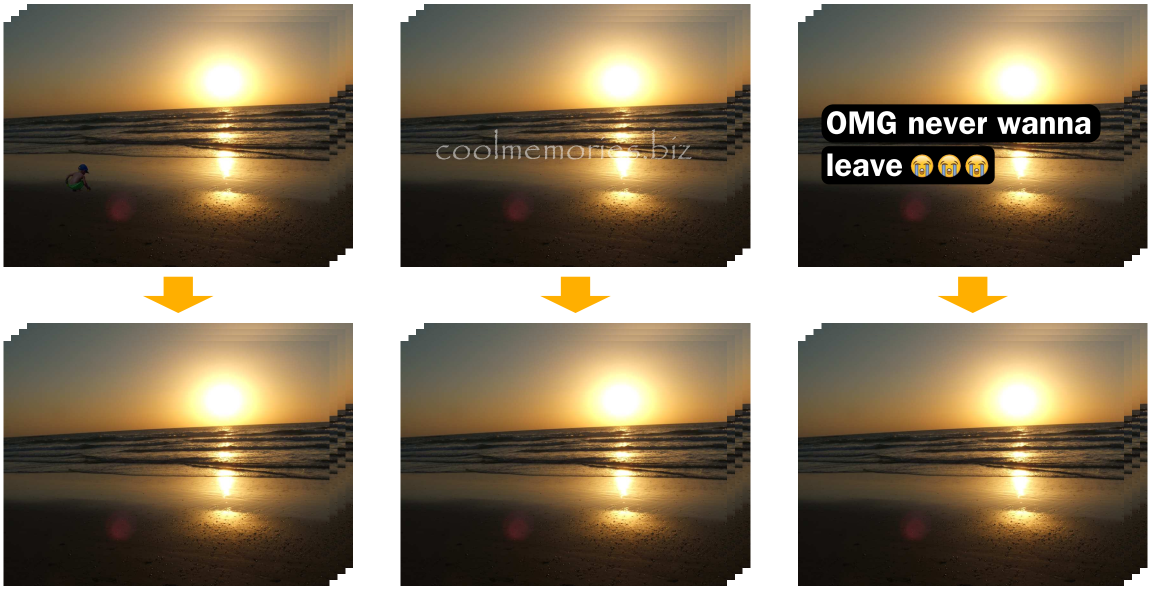
The most advanced video manipulation techniques exist in the form of *generative video models*, which characterize an interesting distribution of videos through a process that can be sampled, typically with some conditioning input. Examples include video interpolation networks, which artificially slow down time [19], and style transfer networks, which make videos look like classical art [20], [21]. Generative video models must be advanced enough to handle a variety of challenges, e.g., reasoning over a complex spatiotemporal space and generating realistic structure in said space. In recent years, most generative visual modeling research has revolved around a particular set of core tasks, including but not limited to generation from scratch [22]–[24], super-resolution [25]–[28], style transfer [20], [21], frame interpolation [19], [29], [30], and generalized video-to-video translation [31], [32].

1.2 Video Inpainting

This dissertation focuses on the core task known as *video inpainting* [33]–[36]. At a high level, video inpainting methods aim to replace a certain missing portion of a video with sufficiently "plausible" values, meaning that the replacement should cause the resulting video to become indistinguishable from a real one. The missing portion of the video, a.k.a. the masked region, may cover a small or large part of the video across both spatial dimensions (i.e., width and height) and the temporal dimension.

Video inpainting has major applications in social media, film, and beyond [33], [35], [37]. As an illustrative example, imagine a picturesque sunset on a beach with calm, soothing waves rolling in and out. It is the perfect scene that makes you want to pull out your phone and film it to preserve the memory. You press the record button, but right after doing so, a young boy runs into view and starts playing around in the sand. By the time he gets out of the way, the sun has already set, and all you have on your phone is a recording of a naïve boy to haunt your memories. Given labels of where the boy is in the video, a video inpainting method can replace him with the background, restoring the memory that you wanted to capture and allowing you to share it with friends without embarrassment (Figure 1.3a).

Object removal is just one of many video inpainting applications for social media; it can also be used for watermark and caption removal. For example, suppose you upload your sunset video to a video hosting website, and then delete the original to save space on your phone. But then your relative, who has not set up an account for that site, asks for a copy. You download the remote copy,



(a) Object removal (b) Watermark removal (c) Caption removal

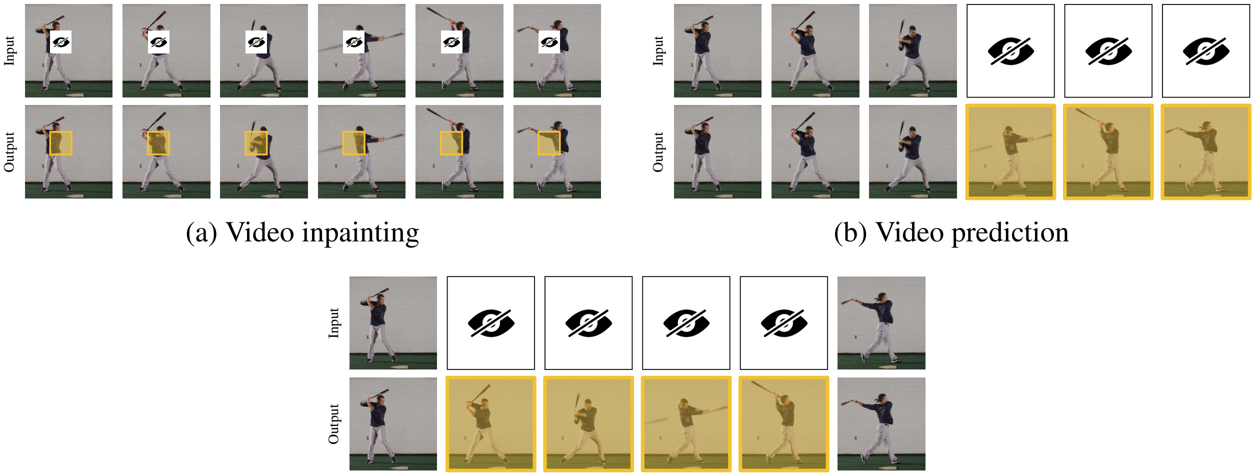
Figure 1.3: Video inpainting applications.

only to find that the website has overlaid a huge watermark that ruins the moment. In this case, video inpainting would let you remove the watermark and share the moment with your relative as intended (Figure 1.3b). Similarly, videos shared on social media often include custom captions that are meant to be funny, but actually backfire. In this case, video inpainting can remove the offending caption and restore the original content (Figure 1.3c).

Furthermore, video inpainting can be used to produce useful research data for other computer vision tasks, particularly when foreground objects distract from the desired downstream task. For instance, imagine a roboticist who wants to create a 3D model of the static parts of a work environment for motion planning purposes; additionally, imagine that they only have access to RGB cameras to capture the scene, and that physically clearing the room of dynamic objects is prohibitively expensive (e.g., due to heavy objects or the potential disruption of daily activities). Video inpainting would allow the roboticist to remove dynamic objects from the captured scene, facilitating a reconstruction of the static environment (via Structure from Motion) with minimal physical effort.

1.2.1 Problem Statement

We formally define the video inpainting task as follows. Let $\mathcal{X}$ be a set of possible values for a pre-determined color space (e.g., $\{0, 1, \ldots, 255\}$ for RGB), and let $V \in \mathcal{X}^{H \times W \times C \times T}$ be an input video with a resolution of $H \times W$ pixels, C color channels, and T frames. V contains a placeholder value for missing voxels (i.e., spatio-temporal pixels) whose locations are indicated by the occlusion



(a) Video inpainting

(b) Video prediction

(c) Frame interpolation

Figure 1.4: Visualization of video inpainting and related tasks. The input row corresponds to the input video V, and the white area corresponds to the placeholder values and the occlusion mask M. The yellow area corresponds to the desired inpainting prediction $f^*(V, M)$.

mask $M \in \{0, 1\}^{H \times W \times T}$. Additionally, let score_{vq} denote an abstract function that quantifies the visual quality of a video given an auxiliary information context A (both score_{vq} and A are defined concretely in Section 1.2.3). The goal of video inpainting is to determine, given V and M, the function f^* yielding values that, when used to inpaint V, maximize the visual quality of the result:

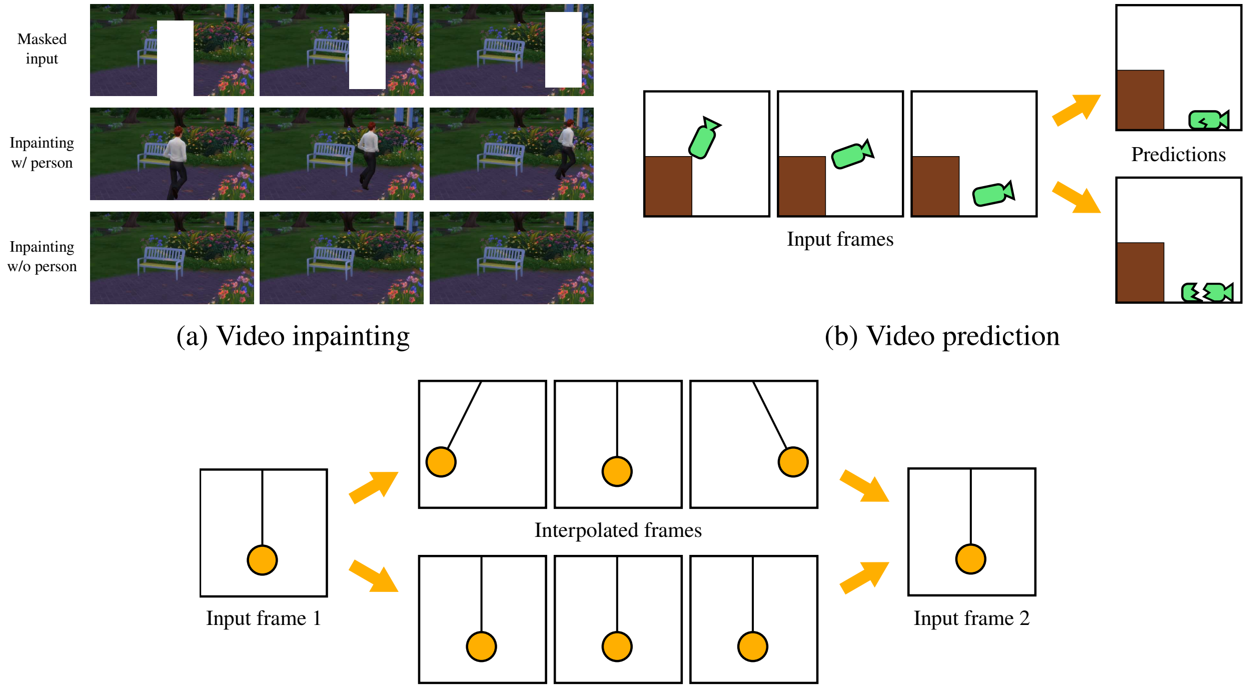
$$f^* = \arg\max_f \ \text{score}_{\text{vq}}\big(f(V, M), A\big) \tag{1.1}$$

Video inpainting is typically studied as a reconstruction problem. Specifically, input video V is generated by taking a real-world source video—denoted as V^*—and masking out the voxels of M. As described in Section 1.2.3, V^* is usually required to compute the visual quality of $f(V, M)$.

We visualize the video inpainting task in Figure 1.4a. Constrained formulations of video inpainting have been studied under different names—for instance, *video prediction* aims to produce a sequence of frames that follow an input sequence (Figure 1.4b), and *frame interpolation* aims to produce one or more frames that depict the progression between two input frames (Figure 1.4c). With respect to Equation 1.1, these formulations correspond to forcing $M_{i,j,k}$ to be 1 for all i, j if k is in a set of missing time steps $\tau \subset \{0, \dots, T-1\}$.

1.2.2 Semantic Ambiguity

It is worth noting that f^* as introduced in Equation 1.1 is not well-defined, i.e., the video inpainting problem does not establish any particular f as the only correct solution. This leads to the primary challenge of video inpainting: *semantic ambiguity*. In other words, there are many



(a) Video inpainting

(b) Video prediction

(c) Frame interpolation

Figure 1.5: Rhetorical visualizations of how semantic ambiguity can arise in video inpainting and related tasks. (a) The masked region can be inpainted with or without a person; both options are equally sensible. (b) The input sequence can lead to the vase breaking or not. (c) The two input frames can result from the pendulum swinging or standing still.

plausible ways to fill in the missing area, each arising from a different explanation of the events in the scene observed through V and M. For example, given a video recorded in a public park, it would be reasonable to fill in the missing area either with a continuation of the background or an object that typically appears in a park, such as a person or a bench (Figure 1.5a).

In constrained formulations of video inpainting, other forms of semantic ambiguity become more prominent. For example, for video prediction, challenges arise from many plausible futures—for instance, in Figure 1.5b, the vase might or might not break. For frame interpolation, aliasing can lead to multiple explanations of the input, such as in Figure 1.5c, where the pendulum could be moving or standing still.

Semantic ambiguity raises challenges both for solution design and for evaluation. In terms of solution design, it is important for video inpainting models to leverage the available information in a way that eliminates spurious predictions. Two broad approaches include reasoning about uncertainty explicitly within the model [38]–[40] and augmenting the given information in some way (as discussed in Section 1.3, the latter approach constitutes a major thrust of our work). As for evaluation, it is worth considering alternative schemes that do not penalize inpainting predictions for deviating from an assumed ground truth. Concretely, this suggests that the reconstruction-based formulation of video inpainting should be evaluated not only with reconstruction-based metrics, but also with metrics that measure complementary properties like realism and temporal consistency (Section 1.2.3).

1.2.3 Measuring Visual Quality

Delving further into the visual quality function, score_{vq} depends on three complementary aspects—*realism*, *temporal consistency*, and *reconstruction performance*—each of which depends on its own sources of auxiliary information (A is the union of all auxiliary information). We express this statement as a weighted sum of three individual functions $\text{score}_{\text{real}}$, $\text{score}_{\text{temp}}$, and $\text{score}_{\text{recons}}$, each of which consumes the inpainting result $f(V, M)$ and its own auxiliary context A_*:

$$\begin{aligned} \text{score}_{\text{vq}} = \lambda_{\text{real}}\text{score}_{\text{real}}\big(f(V, M), A_{\text{real}}\big) + \lambda_{\text{temp}}\text{score}_{\text{temp}}\big(f(V, M), A_{\text{temp}}\big) \\ + \lambda_{\text{recons}}\text{score}_{\text{recons}}\big(f(V, M), A_{\text{recons}}\big) \end{aligned} \quad (1.2)$$

Each λ_* is a constant denoting the weight of one visual quality aspect. Furthermore, the source video from which the video inpainting input V is derived, V^*, is considered a component of auxiliary information A within our formulation. Additional examples of auxiliary information are discussed in Chapter 2.

We now provide an overview and motivation for each of the three aspects of visual quality (technical details are available in Chapter 2). First, realism captures how well an inpainted video fits within the distribution of real videos. It is often measured by comparing the statistics of DNN features between artificial and real videos, e.g., with the Fréchet Inception Distance [41]. Realism best captures the desired goal of plausibility since its generality imposes minimal constraints on the inpainted video. However, modeling the distribution of real videos, even discriminatively, is extremely challenging; for this reason, it is useful to measure complementary aspects of visual quality.

Meanwhile, temporal consistency measures the change in color among pixels that correspond to the same location in the captured scene across time. Techniques to quantify temporal consistency include comparing colors at the endpoints of optical flow estimates [42] and scoring patch similarity across frames [21]. These measurements leverage the fact that real-world scenes, as well as camera motion, tend to change gradually with respect to standard video frame rates, i.e., 24-30 frames per second (FPS). Low temporal consistency manifests in the form of flickering artifacts, which simultaneously distract viewers and give away the artificial nature of the video inpainting result.

Finally, reconstruction performance penalizes any deviations between the inpainted video, $f(V, M)$, and the source video from which V was derived, V^*. Corresponding frames of inpainted and source videos can be compared via pixel-wise differences (e.g., Mean Square Error and Peak

Signal-to-Noise Ratio), semantic feature distances (e.g., the Learned Perceptual Image Patch Similarity metric [43]), or intermediate metrics (e.g., Structural Similarity [44]). Reconstruction performance reflects the standard reconstruction-based formulation of video inpainting, and is the least ambiguous aspect of visual quality discussed in this dissertation (the solution that maximizes it is self-evident and non-degenerate). However, as discussed in Section 1.2.2, it fails to capture the ill-defined nature of video inpainting, hence the need for complementary metrics.

1.3 Contributions

The contributions of this dissertation revolve around two aspects of video inpainting: *temporal context* (Section 1.3.1) and *diagnostic evaluation* (Section 1.3.2). Temporal context refers to relevant information from neighboring frames, and is important for video inpainting because it provides semantic information that distinguishes plausible solutions from implausible ones. Meanwhile, diagnostic evaluation refers to techniques that elucidate when video inpainting models fail, how failures manifest in the output, and what properties of a given model lead to failures. Such evaluation strategies are valuable because they provide stronger guarantees on when video inpainting models can be expected to perform well.

1.3.1 Temporal Context

To inpaint a given video frame, a video inpainting model does not need to rely solely on information within that frame; instead, it can leverage information from neighboring frames, which forms the *temporal context* for the task. Temporal context provides two benefits to inpainting models. First, it reduces the ambiguity of the task by providing semantic constraints. For instance, in Figure 1.5c, the set of possible intermediate states of the pendulum can be reduced by providing additional input frames. Second, temporal context provides appearance information that can be replicated in the target frame due to the redundant nature of video frames.

In this dissertation, we propose two methods that exploit the benefits of temporal context. Our first method, the Bidirectional Temporally-Aware Interpolation Network (bi-TAI), predicts several contiguous full frames using a wider temporal context than prior work. In particular, prior full-frame solutions could only consume one frame from both sides of the desired sequence, or multiple frames from one side; on the other hand, ours leverages multiple frames from both sides. By leveraging an extended temporal context, bi-TAI successfully reduces semantic ambiguity on four video classification datasets, thereby producing results with better visual quality and semantic agreement with the input data.

Our second method, Hyperconsistency (HyperCon), applies frame-wise models to videos in a temporally consistent manner by synthesizing and exploiting temporal context between input frames.

Specifically, it identifies temporally consistent components within a processed, frame-interpolated version of the input video, and then transfers these consistencies to the final output at the desired frame rate. Unlike prior work, our method improves temporal consistency without introducing discolorations; it also exhibits stronger performance on tasks as disparate as video inpainting and style transfer.

1.3.2 Diagnostic Evaluation

In early video inpainting work, researchers relied on individual qualitative examples to analyze their proposed methods; as a result, it was difficult to judge their efficacy in broad, real-world scenarios. However, researchers have more recently demonstrated their methods on larger video datasets, e.g., DAVIS [45] and YouTube-VOS [46], and have leveraged quantitative metrics such as Learned Perceptual Image Patch Similarity [43] and Fréchet Inception Distance [41] to summarize visual quality, thereby providing more compelling evidence of robust performance. Quantitative evaluation on large-scale datasets is quickly becoming the norm in video inpainting, bringing it closer in line with other popular computer vision tasks.

Despite these advances, most video inpainting evaluation focuses on demonstrating strong performance over an entire dataset without thoroughly analyzing how and when a given method fails. For instance, can an inpainting method reconstruct the appearance of an object that has never appeared during training? Can it extrapolate a part of the background that is never visible throughout the video? The answers to behavior-oriented questions such as these remain poorly understood, primarily due to the lack of sufficient quantitative evaluation under fine-grained failure modes.

Our diagnostic evaluation techniques address this gap in knowledge by highlighting failure modes and quantifying failure along orthogonal, interpretable axes. We propose two such solutions: the Moving Symbols software library (Chapter 5) and the DEVIL benchmark (Chapter 6). Moving Symbols allows researchers to produce synthetic video datasets governed by easily modifiable appearance and motion statistics; furthermore, objects in Moving Symbols videos have accompanying metadata to enable a fine-grained analysis of video prediction performance. Using this software, we demonstrate that a state-of-the-art video prediction model fails to extrapolate appearance and motion information—despite its presence in the input sequence—and instead relies on the memorization of priors from the training set.

Meanwhile, the DEVIL benchmark is a comprehensive suite of evaluation tools that illuminate the strengths and weaknesses of modern video inpainting methods at scale. The available videos target failure modes specific to video inpainting, namely those induced by the camera and background scene motion in RGB videos, as well as the motion and size of occlusion masks. Moreover, the evaluation scheme comprehensively summarizes performance under several failure modes with metrics that assess multiple aspects of visual quality. The DEVIL benchmark allows us to objectively compare the behaviors of seven state-of-the-art inpainting models in the largest study of its kind to date.

1.4 Thesis and Impact Statement

Through innovations in temporal context and diagnostic evaluation, our work enforces a higher standard of visual quality in video inpainting results on a larger scale than prior work. Specifically, we propose novel algorithmic and diagnostic techniques that illuminate the relationship between visual quality and video inpainting model design. We assess visual quality along three orthogonal axes—realism, temporal consistency, and reconstruction performance—and propose algorithms that demonstrably improve performance by exploiting temporal context. Furthermore, we introduce evaluation tools that highlight video inpainting failure modes and quantify their impact on visual quality at scale in a fine-grained manner.

The impact of our work is two-fold. First, our techniques enable researchers and ordinary consumers to produce more convincing manipulated videos for their data processing and social media needs, as well as to understand the positive and negative consequences of video manipulation. Second, our diagnostic tools elevate standards for future video inpainting evaluation by facilitating analysis that is fine-grained, replicable, open, and directly comparable between video inpainting methods.

CHAPTER 2

Related Work

2.1 Datasets and Reconstruction

Video inpainting is generally studied as a reconstruction problem: masked input videos are produced by masking out certain values of regular RGB videos, and the goal is to recover the original masked values. Since this dissertation relies on large-scale evaluation and reconstruction data samples, we provide a detailed explanation of the techniques used to generate reconstruction samples at scale. We mainly consider the context of deep learning methods, which have provided the deepest exploration of this topic among prior work.

Within deep learning methods, the reconstruction formulation of video inpainting lends itself to a self-supervised approach, i.e., the supervised data used to train the inpainting model are generated automatically without expert annotations. Because deep neural networks (DNNs) are sensitive to training data [47], [48], the structure of the mask can greatly affect their performance at test time. It is possible to apply simple dropping schemes such as uniform selection (as is standard for general tensor completion methods); however, more sophisticated schemes are needed to simulate the primary applications of video inpainting (i.e., object and watermark removal) and therefore improve DNN model performance under real-world settings. Typically, such schemes produce masks of contiguous regions that resemble a static or moving object in the video.

To train and evaluate video inpainting DNNs, most researchers source videos from large datasets such as the Caltech Pedestrian Detection dataset [49], FaceForensics [50], YouTube-VOS [51], and the DAVIS video object segmentation dataset [45]. However, Lee et al. [52] and Oh et al. [53] adopt alternative strategies during training: specifically, they extract a random set of frames from self-selected YouTube videos, or apply a sequence of random affine transformations on single images from the Places dataset [54].

As for simulating object masks, researchers have employed a wide variety of techniques. One simple option is to randomly mask out a rectangular region within the frame, and either update the size and position at every time step or keep them fixed [55], [56]. Another approach is to utilize the irregular mask dataset from Liu et al. [57], originally proposed to evaluate image inpainting methods; Kim et al. [56] construct video masks from this dataset via random transformations at each time step. Yet another technique is to generate moving objects in a procedural manner as done by Chang et al. [37], [58]. In particular, they manipulate strokes defined by a sequence of control points, where each stroke varies in width and each control point randomly moves based on its velocity and acceleration.

In addition to purely automated strategies, video inpainting masks can also be produced in ways that utilize real data. One such option is to copy real shapes from an object database such as the MIT Saliency Benchmark [59] or the PASCAL VOC 2012 challenge [60]; Lee et al. [52] and Oh et al. [53] opt for this approach by overlaying shapes onto the video frame and translating them over time. Another option, adopted by Kim et al. [56], is to copy foreground object masks from a video object segmentation dataset such as YouTube-VOS [51]. Although segmentations must be annotated manually, they provide more realistic foreground object shapes than automatic approaches.

2.2 Evaluation

As motivated by Section 1.2.3, we evaluate visual quality in this dissertation along three orthogonal axes, following prior video inpainting work [37], [53], [56], [61], [62]. These axes are realism, temporal consistency, and reconstruction performance, which are described in the remainder of this section.

Realism The Fréchet Inception Distance [41] (FID) is a realism metric that has been popularized by image generative adversarial network literature. FID is computed by first fitting multivariate normal distributions over samples of deep neural network representations of artificial and real images, and then computing the Fréchet distance between these distributions [63]. In video inpainting literature, FID has been adapted to consume videos instead of images [37], [53], [56]—in this case, the artificial videos correspond to inpainted videos, and the real videos correspond to those from which the masked videos to inpaint are derived. In the context of Equation 1.2, the auxiliary information A_{real} encapsulates the distribution of real videos.[1]

Temporal Consistency One way to compute temporal consistency, assuming that the optical flow of a given video is known, is to take the mean difference of colors of corresponding points between consecutive frames. In blind video consistency work, optical flow is extracted by applying a flow estimation model on the unprocessed video [42]. Likewise, in video inpainting, optical flow is estimated from the source of the masked video, V^* [62]. Note that collecting ground-truth optical

[1] Video FID is slightly inconsistent with our formulation of plausibility in Equation 1.2 since it consumes sets of inpainted videos rather than individual ones; regardless, we retain our formulation for pedagogical clarity.

flow in real videos requires specialized capture setups, and is thus generally unavailable. For this reason, there exist other temporal consistency metrics that leverage the redundancy of video frames without assuming that optical flow is given. For instance, the patch consistency metric proposed by Gupta et al. [21] randomly samples a patch from one frame and computes the best similarity score between it and neighboring patches in the next frame.

Reconstruction Performance The most straightforward way to measure reconstruction performance is by computing an established image similarity metric between corresponding frames, e.g., Peak Signal-to-Noise Ratio, Structural Similarity [44], or Learned Perceptual Image Patch Similarity (LPIPS) [43]. However, extensions of these metrics to contiguous video frames are also possible, and are capable of summarizing high-level differences that occur on a wider time scale. For instance, LPIPS may be modified to extract video clip features from a 3D convolutional neural network such as I3D [64]; we adopt this approach in our DEVIL benchmark (Chapter 6).

2.3 Methods

In this section, we provide an overview of the various classes of video inpainting approaches:

- *General tensor completion methods*, which find values that minimize the rank of the inpainted video tensor (Section 2.3.1);
- *Object-based methods*, which repair foreground and background objects by exploiting their typical behaviors (Section 2.3.2);
- *Repetitive tensor-based methods*, which capitalize on the repetitive nature of videos to borrow suitable patches from the known region (Section 2.3.3); and
- *Deep learning methods*, which train deep neural networks (DNNs) to produce inpainting values (Section 2.3.4).

2.3.1 General Tensor Completion

Given that videos are tensors with specific dimensions, video inpainting can be viewed as a type of tensor completion problem, for which several solutions have been proposed. General tensor completion methods—general in that the tensors need not represent any specific type of data—assume that the completed tensor has a low rank, which is done to make the problem well-posed and tractable. Specifically, given incomplete tensor $\mathcal{T}$, they aim to find the complete tensor $\mathcal{X}$ with minimal rank whose entries are consistent with $\mathcal{T}$:

$$\begin{aligned} \underset{\mathcal{X}}{\text{minimize}} \quad & \text{rank}_*(\mathcal{X}) \\ \text{subject to} \quad & \mathcal{X}_\Omega = \mathcal{T}_\Omega \,. \end{aligned} \tag{2.1}$$

Ω denotes observed entries, and rank$_*$ is a rank operator chosen from assumptions made on $\mathcal{T}$.

One major class of general tensor completion methods focuses on minimizing the tensor trace norm, i.e., a high-dimensional generalization of the matrix trace norm. These approaches are inspired by the fact that the matrix trace norm provides the tightest lower bound on the matrix rank obtainable by convex methods [65]. In particular, Liu et al. [66] define the tensor trace norm $\|\mathcal{X}\|_*$ as a convex combination of trace norms of $\mathcal{X}$ unfolded along each dimension:

$$\|\mathcal{X}\|_* \coloneqq \sum_i \alpha_i \|X_{(i)}\|_* \,, \tag{2.2}$$

where $X_{(i)}$ is the matrix obtained by unfolding $\mathcal{X}$ along dimension i, and the α_is are nonnegative coefficients that sum to 1. From this definition, they propose three ways to minimize $\|\mathcal{X}\|_*$, one that adds a smoothness term and two that use non-smooth descent algorithms. Mu et al. [67] improve efficiency by reshaping the unfolded matrices to be more square while preserving their low-rank properties.

The other major class of tensor completion methods relies on decomposition. Rather than solving for the missing values directly, they decompose the tensor into a product of several simpler elements with pre-determined maximum ranks; the problem is then reduced to determining the simpler elements from the incomplete tensor. The primary decompositions used in prior work include the CANDECOMP/PARAFAC (CP) decomposition [68], which expresses a tensor as the outer product of several vectors; and the Tucker decomposition [69], which uses a core tensor multiplied by matrices along each dimension. Representative approaches include Zhang et al. [70], who utilize the Tensor Singular Value Decomposition (t-SVD) [71] to perform video completion and denoising; and Kasai [72], who proposes a CP decomposition-based algorithm for 3D tensors in which one dimension grows over time (i.e., streaming video).

The aforementioned general tensor completion methods have been demonstrated in small video inpainting experiments as proofs-of-concept. However, they are not well-suited for real-world applications, such as object and watermark removal, due to their assumption that unmasked entries are uniformly distributed. Because objects and watermarks consist of several contiguous pixels per frame, this assumption is usually violated in practice. Furthermore, the amount of storage required for intermediate computations can be prohibitively large, especially for long, high-resolution videos [73]. For more information on general tensor completion methods, we refer the reader to the excellent survey by Song et al. [74].

2.3.2 Object-Based Methods

Among the earliest methods specifically designed for video inpainting, it is common to exploit typical behaviors of foreground objects and the background to repair them in a tractable manner. For example, Zhang et al. [75] stitch background elements into several reference layers, and then warp the reference layers back into each video frame. Similarly, Jia et al. [76] construct background templates and paste them into each frame, and additionally reconstruct foreground objects by modeling and completing their periodic motion. Patwardhan et al. [77] take an alternative approach by first repairing the moving foreground, and then the background as subsequent processes. Specifically, they repair the foreground by searching for and copying fragments guided by a foreground mosaic/stitch image; then, they repair the remaining content by aligning and copying pixels from neighboring frames. Jia et al. [78] do not explicitly divide foreground and background recovery into separate processes; however, they exploit object motion through mean-shift tracking, which is used to prioritize missing patches on the edge of the missing region and to locate suitable source patches to copy into the missing region. These methods make strong assumptions about foreground and background appearance within the video—e.g., repetitive foreground motion, consistently-ordered background layers, and consistent object appearance and illumination across frames—and are thus applicable only in limited use cases.

More recent object-based methods relax the aforementioned assumptions by leveraging piece-wise homographies. For instance, Granados et al. [79] align foreground and background layers across frames with piece-wise homographies, copy pixels from the aligned frames, and then post-process the result to remove inconsistent illumination. Ebdelli et al. [80] follow a similar approach, using pixel-wise weighted homographies for alignment and Poisson blending [81] for post-processing. Although these methods have improved performance and increased the number of applicable use cases, they rely on strong homography estimates, which may be difficult to obtain on untextured or indistinguishable backgrounds; furthermore, they fail to account for foreground motion near masked regions.

2.3.3 Repetitive Tensor-Based Methods

Repetitive tensor-based methods for video inpainting assume that spatiotemporal patches are repeated often throughout the inpainted video, and leverage this assumption to either model or directly borrow from the unmasked region in pursuit of the masked region. They differ from general tensor completion methods in that they aim to recover contiguous missing regions given the availability of contiguous observed regions, forgoing the assumption of uniform random missing data. They also contrast with object-based methods in that they make no assumptions about the foreground and background content of the video, theoretically enabling better performance when the typical assumptions for object-based methods are broken.

It is possible to express the completed video with a probabilistic model—for instance, Cheung et al. [82] learn video epitomes that express the statistics of the video with a 3D grid of multivariate normal distributions from which inpainted values can be sampled. However, it is more common for repetitive tensor-based methods to explicitly borrow elements from unmasked parts of the video by solving an objective that encourages consistency among borrowed elements and/or their neighborhoods. For example, Granados et al. [33] copy voxels into the inpainted region such that adjacent voxels in the inpainted region have similar source neighborhoods.

Repetitive tensor-based methods operate more frequently at the patch level than at the voxel level. Patch-level techniques have largely been inspired by Wexler et al. [83], who induce global consistency by promoting local consistency among spatiotemporal patches that overlap a voxel in the missing region. Subsequent patch-based methods have focused on improving search features and speed; for instance, Newson et al. [34] expedite search by extending the image-based PatchMatch algorithm [84] to spatiotemporal patches, and Huang et al. [35] incorporate optical flow and geometric transforms to further improve the quality of borrowed patches. Although repetitive tensor-based techniques can perform well without strict assumptions on the foreground and background content, they can perform poorly under large camera motion and changes in illumination.

2.3.4 Deep Learning

Deep learning methods for video inpainting estimate a mapping between incomplete videos and their completed counterparts by fitting a deep neural network (DNN) to self-supervised data. Although most deep learning methods rely on self-supervision from a held-out training set, there exist exceptions; for example, Zhang et al. [36] train a frame-wise 2D CNN strictly on the masked video to be inpainted. They utilize losses that encourage the predicted frames and the optical flow maps to match corresponding information that is available within the masked video.

Prior approaches can be subdivided into several classes, where each class encourages certain behaviors of the model through carefully-designed network architectures. For instance, one class adapts traditional image autoencoder architectures (e.g., UNet [85]) for use on masked videos; in particular, they utilize 3D convolutional neural networks (CNNs), applying tweaks to manage the explosion of data from the additional time dimension and reduce the total number of floating-point operations (FLOPS). Chang et al. [37], for example, implement an autoencoder with gated 3D convolutional filters whose outputs strictly depend on unmasked values from their inputs; they further extend this model in [37] with semi-separable 3D kernels, allowing it to maintain a similar level of performance with significantly fewer parameters and FLOPS. Wang et al. [55] propose a stacked autoencoder network that first processes the video at a downsampled resolution with a 3D CNN, and then refines the result frame-wise with a 2D CNN.

Meanwhile, flow-based deep learning methods predict optical flow to either borrow pixel values from the unmasked region directly or modulate intermediate features in an interpretable manner. For example, Xu et al. [61] predict the optical flow of the completed video in a coarse-to-fine manner (i.e., at three gradually-increasing resolutions), and then propagate unmasked pixels via the predicted flow. Kim et al. [56] propose a recurrent model that generates intermediate flow maps, which are used to align features from neighboring frames to the current predicted frame.

Yet another class of deep learning methods provides explicit mechanisms for copying intermediate features from distant frames. For instance, Lee et al. [52] align reference frames to the target frame via affine transformations that are estimated with an alignment network; then, they copy features from the reference frames at corresponding spatial locations using self-attention. Oh et al. [53] train a model to select values from reference frames based on the similarity between encoded reference keys and target queries; this removes the need to find explicit global affine alignments between frames.

2.3.4.1 Losses

Loss functions play a key role in training deep learning models to perform specific tasks. The majority of those used for video inpainting quantify the similarity between the inpainted video $\hat{V} = f(V, M)$ and a corresponding "ground-truth" video V^* from which input V is derived (per the notation from Chapter 1, V is an RGB video with placeholder values at missing voxels, M is the mask indicating the missing voxels of V to inpaint, and f is the inpainting model).

Mean absolute/square error The most common losses among deep video inpainting methods are the mean absolute/square error (MAE/MSE), which compare the values between the inpainted and ground-truth videos at corresponding spatial locations across all frames. Let H, W, T denote the height, width, and number of frames of the input video respectively. MAE/MSE are then defined as:

$$\mathcal{L}_{MAE} = \frac{\sum_{h,w,t} \| V^*_{h,w,t} - \hat{V}_{h,w,t} \|}{HWT}, \tag{2.3}$$

$$\mathcal{L}_{MSE} = \frac{\sum_{h,w,t} \| V^*_{h,w,t} - \hat{V}_{h,w,t} \|_2^2}{HWT}, \tag{2.4}$$

where h, w, and t index a voxel in the video. Different weights can be assigned to the masked and unmasked parts of the video, as is done in [52], [53], [55].

Perceptual loss Originally developed for image super-resolution and style transfer [86], the perceptual loss compares the features between corresponding inpainted and ground-truth frames from multiple levels of an image classification network. Let $\hat{I}_t$ and I_t^* respectively denote a frame of the inpainted video and its corresponding ground truth. Also, let $\phi_l(x)$ denote the operation that extracts features from layer l of an image classification network given an arbitrary image x, and let $\mathbb{L}$ be the set of layer indexes from which features are extracted. Furthermore, let T be the number of frames in the video, and H_l, W_l, C_l be the dimensions of feature activations from layer l. The perceptual loss is defined as:

$$\mathcal{L}_{perc} = \frac{1}{T} \sum_{t=1}^{T} \sum_{l \in \mathbb{L}} \frac{1}{H_l W_l C_l} \|\phi_l(I_t^*) - \phi_l(\hat{I}_t)\|_2^2 \,. \tag{2.5}$$

Video inpainting methods that use perceptual loss include Lee et al. [52], Chang et al. [37], [58], and Zhang et al. [36].

Style loss Adapted from the style transfer approach of Gatys et al. [87] and used in [37], [52], [58], the style loss compares the correlations between features in corresponding frames by comparing Gram matrices defined on feature activation tensors. Specifically, let $G_l^{\phi}(x)$ denote a Gram matrix defined by the feature activations of an image classification network at layer l given an arbitrary image x. $G_l^{\phi}(x)$ is computed by unfolding the feature activation tensor $\phi_l(x)$ along the spatial dimensions—thereby producing a matrix $\psi_l(x)$ of size $C_l \times H_l W_l$—and then multiplying $\psi_l(x)$ by its transpose:

$$G_l^{\phi}(x) = \psi_l(x) \psi_l(x)^{\top} \,. \tag{2.6}$$

The style loss is the scaled MAE between Gram matrices of corresponding inpainted and ground-truth frames summed across all layers and time steps:

$$\mathcal{L}_{\text{style}} = \frac{1}{T} \sum_{t=1}^{T} \sum_{l \in \mathbb{L}} \frac{1}{C_l C_l} \|G_l^{\phi}(I_t^*) - G_l^{\phi}(\hat{I}_t)\| \,. \tag{2.7}$$

Optical flow loss For models that produce optical flow, e.g., [36], [56], [61], it is typical to supervise the flow against corresponding estimates from the ground-truth video V^*. Given t_a and t_b as two distinct video frame indexes, let $\hat{F}_{t_a \to t_b}$ be the dense flow between them as generated by the video inpainting model, and let $F^*_{t_a \to t_b}$ be the flow estimated between time steps t_a and t_b from V^*, e.g., by FlowNet2 [88]. The optical flow loss is the error between $\hat{F}_{t_a \to t_b}$ and $F^*_{t_a \to t_b}$ across several frame pairs $\mathbb{P}$:

$$\mathcal{L}_{\text{flow}} = \frac{1}{|\mathbb{P}|} \sum_{(t_a, t_b) \in \mathbb{P}} \frac{1}{HW} \| F^*_{t_a \to t_b} - \hat{F}_{t_a \to t_b} \| \,. \tag{2.8}$$

The pairs that make up $\mathbb{P}$ can correspond to both adjacent and distant frames [36], [56].

Warping loss As used by Kim et al. [56] and Zhang et al. [36], the warping loss traces inpainted pixels through the optical flow produced by the video inpainting model, checking that corresponding pixels have similar colors. For target time step t_v and source time step t_u, let $\hat{F}_{t_v \to t_u}$ be the flow from time t_v to time t_u produced by the inpainting model. Also, let $\text{warp}(\hat{I}_{t_u}, \hat{F}_{t_v \to t_u})$ be the result of bilinearly sampling from the inpainted frame at time step t_u, $\hat{I}_{t_u}$, using the displacements given by $\hat{F}_{t_v \to t_u}$. The warping loss is the mean error between the warped source frame $\text{warp}(\hat{I}_{t_u}, \hat{F}_{t_v \to t_u})$ and the corresponding target frame $\hat{I}_{t_v}$ across several source and target pairs given by the set $\mathbb{P}$:

$$\mathcal{L}_{\text{warp}} = \frac{1}{|\mathbb{P}|} \sum_{(t_v, t_u) \in \mathbb{P}} \frac{1}{HW} \| \hat{I}_{t_v} - \text{warp}(\hat{I}_{t_u}, \hat{F}_{t_v \to t_u}) \| \,. \tag{2.9}$$

Generative adversarial network loss The generative adversarial network (GAN) loss adaptively penalizes a video inpainting model by training a discriminator to distinguish real video clips from inpainted ones. The discriminator is updated in alternation with the inpainting model (i.e., the generator) to improve its ability to distinguish real and inpainted videos; meanwhile, the inpainting model is optimized under a total loss that incorporates the discriminator's output alongside other objectives (i.e., task-based losses).

Chang et al. [37], [58] adapt the original cross-entropy formulation from Goodfellow et al. [89] to include video inpainting task-based losses. To be precise, let f and D denote the inpainting and discriminator models. Also, let $\mathcal{R}$ and $\mathcal{F}$ respectively denote the set of real videos and the set of videos generated by the inpainting model f. Finally, let $\mathcal{L}_{\text{task}}(v)$ be the sum of inpainting task-based losses on a video v. The losses used to update f and D are:

$$\mathcal{L}_f = \mathbb{E}_{v \sim \mathcal{F}}[\mathcal{L}_{\text{task}}(v) - \log(D(v)] \,, \tag{2.10}$$

$$\mathcal{L}_{\text{D}} = -\mathbb{E}_{v \sim \mathcal{R}}[\log(D(v))] - \mathbb{E}_{V_{\mathcal{F}} \sim \mathcal{F}}[1 - \log(D(v))] \,. \tag{2.11}$$

CHAPTER 3

Video Frame Inpainting

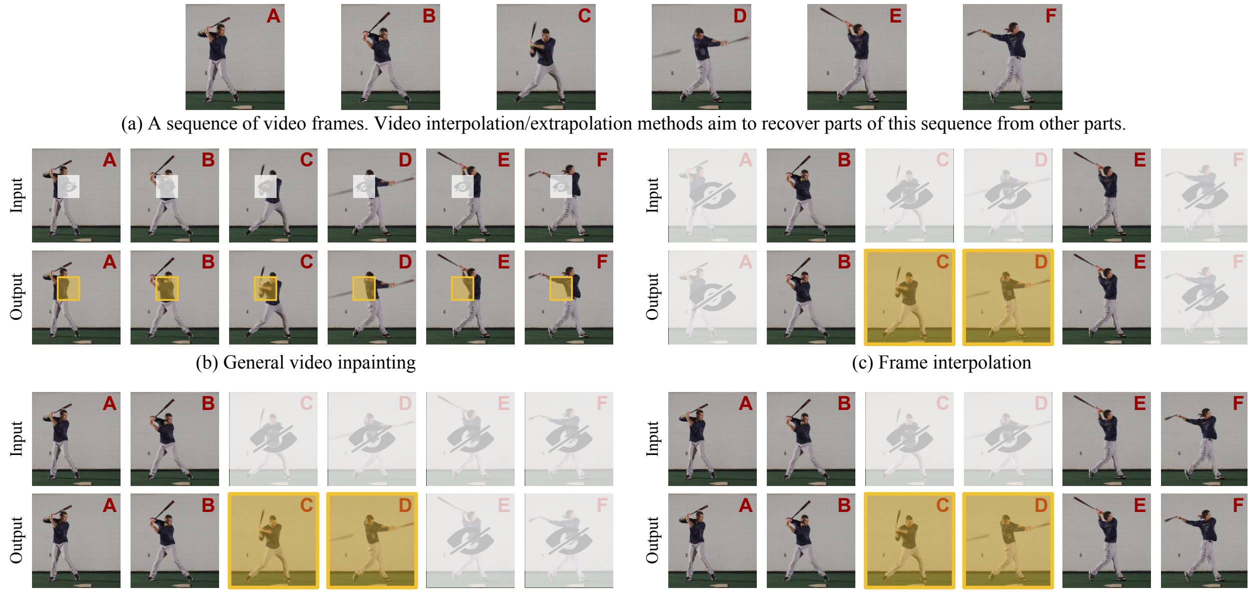
3.1 Introduction

There exist multiple video interpolation/extrapolation tasks where the goal is to synthesize pixels across space and time conditioned on multiple input frames. In particular, three such tasks have received a substantial amount of attention in recent years. In the first task, *general video inpainting*, we are given a video that is missing arbitrary voxels (spatio-temporal pixels), and the goal is to fill each voxel with the correct value. In the second task, *frame interpolation*, the goal is to predict the appearance of one or more frames that lie in between two (typically subsequent) input frames. In the final task, *video prediction*, the goal is to take a sequence of input frames and extrapolate the appearance of multiple future frames.

Unfortunately, these three tasks are limited in terms of their assumptions or well-posed they are. For example, general video inpainting methods are designed to fill in relatively small spatio-temporal regions, and may therefore perform poorly when whole, contiguous frames are missing. Frame interpolation methods fill in whole frames, but they cannot leverage enough contextual information to rule out several plausible predictions (for instance, to predict the appearance of a swinging pendulum, we would need more than two frames to determine its speed). Similarly, video prediction methods cannot leverage information to determine which of several possible futures to predict.

In light of the limitations of these video interpolation/extrapolation tasks, we focus on an underexplored task that lies at their intersection, *video frame inpainting* (shown in Figure 3.1e). In this task, the goal is to reconstruct a missing sequence of middle video frames given contiguous sequences of frames that come immediately before and after it (the *preceding* and the *following* frames). For example, given a clip of someone winding up a baseball pitch and a clip of that person after he/she has released the ball, we would predict the clip of that person throwing the ball. Video frame inpainting methods can be used in multiple applications, e.g. to upsample videos temporally or to smoothly splice video clips taken at different times.

Although our task can be seen as a generalization or modification of other standard video interpolation/extrapolation tasks, ours provides a new combination of challenges that enable method-



(a) A sequence of video frames. Video interpolation/extrapolation methods aim to recover parts of this sequence from other parts.

(b) General video inpainting

(c) Frame interpolation

(d) Video prediction

(e) Video frame inpainting (this work)

Figure 3.1: A visual comparison of various video interpolation/extrapolation tasks. In this paper, we explore (e) video frame inpainting. Unlike general video inpainting methods, we recover whole, contiguous frames; and unlike frame interpolation and video prediction methods, we predict the desired sequence using *multiple* frames that appear both *before* and *after* it.

ological insight. For example, it resembles general video inpainting in the case where the missing voxels are arranged in a certain pattern, but our formulation emphasizes the completion of multiple contiguous frames without ground-truth spatial information at the inpainted time steps, whereas general video inpainting typically emphasizes the completion of regions over a long temporal extent but a limited spatial extent. It also resembles frame interpolation and video prediction with additional input frames, but to the best of our knowledge, prior works in these areas have not leveraged extended temporal context both before and after the desired sequence. Furthermore, compared to frame interpolation and video prediction, the appearance of middle frames is greatly constrained by the extended context on either side, making our formulation more well-defined and reconstruction error more interpretable.

In this work, we present the first deep neural network (DNN) specifically designed for video frame inpainting. Inspired by the recent successes of DNNs for video prediction and frame interpolation, we address the problem in two steps as shown in Figure 3.2. First, we use a video prediction subnetwork to generate two intermediate predictions of the middle frames: the "forward prediction" conditioned on the preceding frames, and the "backward prediction" conditioned on the following frames. Then, we blend each pair of frames corresponding to the same time step to obtain the final prediction. The blending process uses a convolutional neural network (CNN) that interpolates the final frame from the two input frames as well as the corresponding time step in a

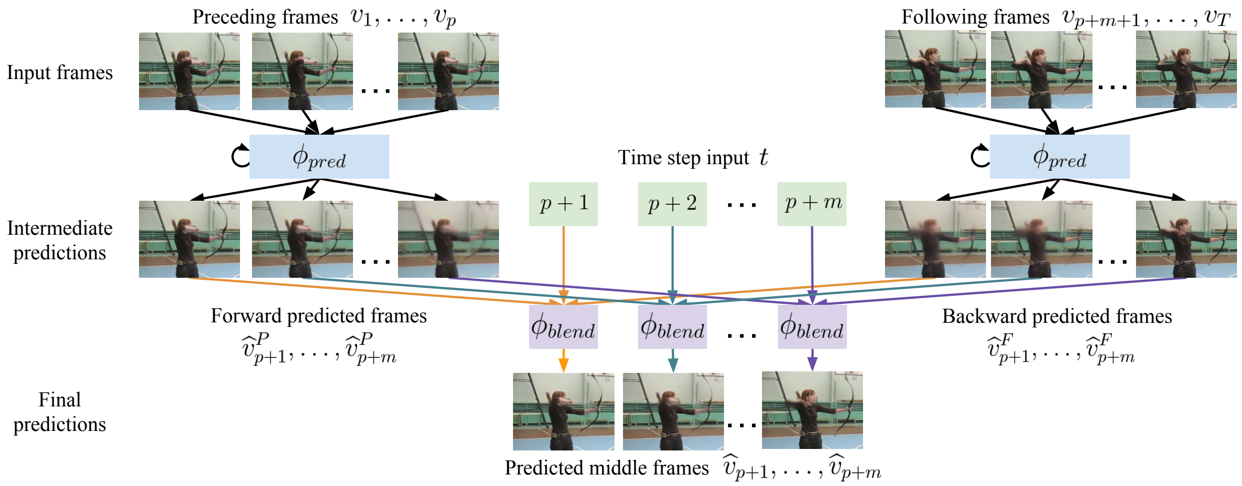

Preceding frames $v_1, \ldots, v_p$
Following frames $v_{p+m+1}, \ldots, v_T$
Input frames
ϕ_{pred}
Time step input t
ϕ_{pred}
$p+1$
$p+2$
$p+m$
Intermediate predictions
Forward predicted frames $\widehat{v}^P_{p+1}, \ldots, \widehat{v}^P_{p+m}$
ϕ_{blend}
ϕ_{blend}
ϕ_{blend}
Backward predicted frames $\widehat{v}^F_{p+1}, \ldots, \widehat{v}^F_{p+m}$
Final predictions
Predicted middle frames $\widehat{v}_{p+1}, \ldots, \widehat{v}_{p+m}$

Figure 3.2: An overview of our bi-TAI method for video frame inpainting. We predict middle frames by blending forward and backward intermediate video predictions (generated by ϕ_{pred}) with a Temporally-Aware Interpolation network (ϕ_{blend}).

data-driven manner. In short, we perform *bidirectional prediction* followed by *temporally-aware interpolation (TAI)* to predict the middle frames; therefore, we name our full method and model **bi-TAI**.[1] As we show in our experiments, bi-TAI yields the most accurate predictions among several state-of-the-art baselines and across multiple human action video datasets.

Blending the intermediate frames generated by the bidirectional prediction process is a non-trivial task; therefore, we develop a TAI strategy that exploits three characteristics of bidirectional prediction. First, a pair of intermediate frames for the same time step might be inconsistent, e.g. an actor might appear in two different locations. To address this, we introduce a CNN for blending/interpolation that modulates the pair of frames with adaptive convolutional kernels and then adds them together; this process can cleanly merge the pair of frames by reconciling the differences between them. Second, for any given time step, the forward and backward predictions are not equally reliable: the former is more accurate for earlier time steps, and the latter is more accurate for later time steps. Hence, we feed time step information directly into the blending network, making it *temporally-aware* by allowing it to blend differently depending on which time step it is operating at. Finally, the intermediate predictions come from a DNN whose hidden features may be useful for blending. To leverage these features, we use them to condition the adaptive convolutional kernels that are used to modulate the intermediate predictions. We implement this strategy with a novel blending CNN called the **Temporally-Aware Interpolation Network** (or TAI network for short), which we describe in Sec. 3.2.4.

In summary, we make the following contributions. First, we propose bi-TAI, a DNN for video frame inpainting that generates two intermediate predictions and blends them together with our

[1] bi-TAI was originally proposed by this dissertation's author in [90], [91].

novel TAI network. Second, we compare our approach to several state-of-the-art methods from the video inpainting, video prediction, and frame interpolation literature, and demonstrate that our method outperforms them quantitatively and qualitatively on several human action video datasets. Finally, we perform ablation studies to demonstrate that using a *temporally-aware* interpolation network (as opposed to simple blending strategies or an interpolation network that does not learn to use temporal information dynamically) is key to blending bidirectional predictions well. We provide an implementation of our model on GitHub.[2]

3.2 Approach

3.2.1 Problem Statement

We define the video frame inpainting problem as follows. Let $V = \{v_1, v_2, \ldots, v_T\}$ be a sequence of frames from a real video, p, m, and f be the number of "preceding", "middle", and "following" frames such that $p + m + f = T$, and $P_V = \{v_1, \ldots, v_p\}$, $M_V = \{v_{p+1}, \ldots, v_{p+m}\}$, $F_V = \{v_{p+m+1}, \ldots, v_T\}$ be the sequences of preceding, middle, and following frames from V respectively. We seek an approximation to the oracle video frame inpainting function ϕ that satisfies $M_V = \phi\left(P_V, F_V\right)$ for all V.

3.2.2 Model Overview

We propose a DNN to approximate the oracle video frame inpainting function ϕ (see Figure 3.2). Our model decomposes the problem into two sub-problems and tackles them sequentially with two modules: the Bidirectional Video Prediction Network (Sec. 3.2.3) and the Temporally-Aware Interpolation Network (Sec. 3.2.4).

- The **Bidirectional Video Prediction Network** generates two intermediate predictions of the middle sequence M_V, where each prediction is conditioned solely on the preceding sequence P_V and the following sequence F_V respectively.

- The **Temporally-Aware Interpolation (TAI) Network** blends corresponding frames from the predictions made by the Bidirectional Video Prediction Network, thereby producing the final prediction $\widehat{M_V}$. It accomplishes this by leveraging intermediate activations from the Bidirectional Video Prediction Network, as well as scaled time steps that explicitly indicate the relative temporal location of each frame in the final prediction.

Even though our model factorizes the video frame inpainting process into two steps, it is optimized end-to-end.

[2] `https://github.com/MichiganCOG/video-frame-inpainting`

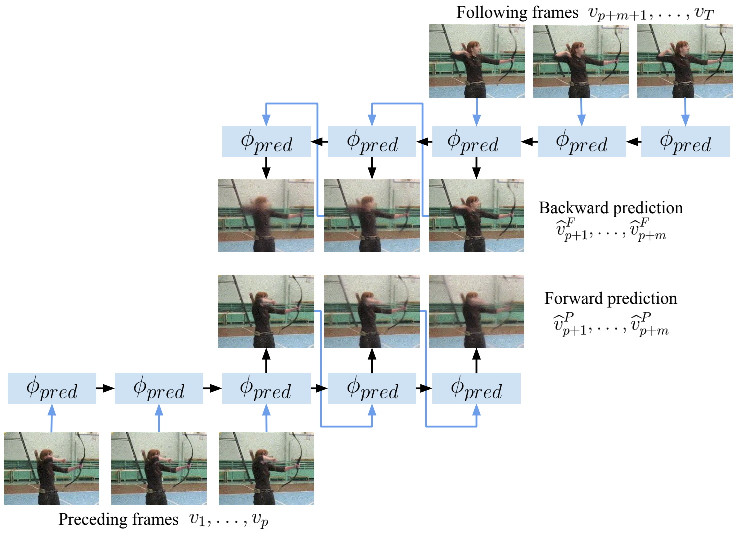

Following frames $v_{p+m+1}, \dots, v_T$
ϕ_{pred}
ϕ_{pred}
ϕ_{pred}
ϕ_{pred}
ϕ_{pred}
Backward prediction
$\widehat{v}^F_{p+1}, \dots, \widehat{v}^F_{p+m}$
Forward prediction
$\widehat{v}^P_{p+1}, \dots, \widehat{v}^P_{p+m}$
ϕ_{pred}
ϕ_{pred}
ϕ_{pred}
ϕ_{pred}
ϕ_{pred}
Preceding frames $v_1, \dots, v_p$

Figure 3.3: The Bidirectional Video Prediction Network.

3.2.3 Bidirectional Video Prediction Network

We use the Bidirectional Video Prediction Network ϕ_{pred}, shown in Figure 3.3, to produce two intermediate predictions—a "forward prediction" $\widehat{M}_V^P = \{\widehat{v}^P_{p+1}, \dots, \widehat{v}^P_{p+m}\}$ and a "backward prediction" $\widehat{M}_V^F = \{\widehat{v}^F_{p+1}, \dots, \widehat{v}^F_{p+m}\}$—by conditioning on the preceding sequence P_V and the following sequence F_V respectively:

$$\widehat{M}_V^P = \phi_{pred}\left(P_V\right) \,, \tag{3.1}$$

$$\widehat{M}_V^F = \left[\phi_{pred}\big((F_V)^R\big)\right]^R \,, \tag{3.2}$$

where $(\cdot)^R$ is an operation that reverses the input sequence. We use the same parameters to generate the forward and backward predictions for two reasons: (i) it substantially reduces the size of the Bidirectional Video Prediction Network, and (ii) forward and backward motion behave similarly in terms of low-level pixel dynamics, so it is beneficial to share the parameters that predict such motion.

The Bidirectional Video Prediction Network recurrently generates each frame by conditioning on all previous frames. For example, for the forward prediction:

$$\widehat{v}^P_{k+1} = \phi_{pred}\big(\{\tilde{v}^P_1, \tilde{v}^P_2, \dots, \tilde{v}^P_k\}\big) \,, \tag{3.3}$$

where for a given t, $\tilde{v}^P_t$ is either v_t (an input frame) if $t \in \{1, \dots, p\}$ or $\widehat{v}^P_t$ (an intermediate predicted frame) if $t \in \{p+1, \dots, p+m\}$. During this phase, we also store a subset of the intermediate activations from the Bidirectional Video Prediction Network, denoted as $\pi_t = \{\pi^1_t, \pi^2_t, \dots\}$, that serve as inputs to the TAI network. We apply an analogous procedure to obtain each frame in the backward prediction $\widehat{v}_t^F$ and its corresponding intermediate activations $\rho_t = \{\rho_t^1, \rho_t^2, \dots\}$.

3.2.4 Temporally-Aware Interpolation Network

Following the Bidirectional Video Prediction Network, the TAI network ϕ_{blend} takes corresponding pairs of frames from $\widehat{M}_V^P$ and $\widehat{M}_V^F$ with the same time step, i.e. $\left(\widehat{v}_t^P, \widehat{v}_t^F\right)$ for each time step $t \in \{p+1, \dots, p+m\}$, and blends them into the frames that make up the final prediction $\widehat{M}_V$:

$$\widehat{v}_t = \phi_{blend}\left(\widehat{v}_t^P, \widehat{v}_t^F\right) , \tag{3.4}$$

$$\widehat{M}_V = \{\widehat{v}_t \mid t = p+1, \dots, p+m\} . \tag{3.5}$$

Blending $\widehat{v}_t^P$ and $\widehat{v}_t^F$ is difficult because (i) they often contain mismatched content (e.g. between the pair of frames, objects might be in different locations), and (ii) they are not equally reliable (e.g. $\widehat{v}_t^P$ is more reliable for earlier time steps). As we show in Sec. 3.3.3, equally averaging $\widehat{v}_t^P$ and $\widehat{v}_t^F$ predictably results in ghosting artifacts (e.g. multiple faded limbs in human action videos), but remarkably, replacing a simple average with a state-of-the-art interpolation network (e.g. Niklaus et al. [92]) also exhibits this problem.

In order to blend corresponding frames more accurately, our TAI network utilizes two additional sources of information. Aside from the pair of frames to blend, it receives the scaled time step to predict, defined as $w_t = (t-p)/(m+1)$, and the intermediate activations from the Bidirectional Video Prediction Network π_t and ρ_t. We feed w_t to the TAI network so it can learn how to incorporate the unequal reliability of $\widehat{v}_t^P$ and $\widehat{v}_t^F$ into its final prediction; we feed π_t and ρ_t to leverage the high-level semantics that the Bidirectional Video Prediction Network has learned, as well as to backpropagate errors through the Bidirectional Video Prediction Network more easily. We contrast standard interpolation with TAI algebraically:

$$\widehat{v}_t = \phi_{interp}\left(\widehat{v}_t^P, \widehat{v}_t^F\right) , \tag{3.6}$$

$$\widehat{v}_t = \phi_{TAI}\left(\widehat{v}_t^P, \pi_t, \widehat{v}_t^F, \rho_t, w_t\right) . \tag{3.7}$$

3.2.5 Network Architecture Details

Our high-level approach to video frame inpainting places few constraints on the network architectures that can be used to implement each module (Sec. 3.2.2). To demonstrate the full potential of our approach, we base the network architectures for each module on the top-performing architectures for video prediction and frame interpolation (to the best of our knowledge as of this writing). We instantiate the Bidirectional Video Prediction Network ϕ_{pred} with MCnet [93] (we review the architecture of MCnet and its use in bi-TAI in Sec. 3.2.5.1). As for the TAI network, we modify the Separable Adaptive Kernel Network [92] to take as input the scaled time step w_t and the intermediate activations π_t and ρ_t (we elaborate on this extension in Sec. 3.2.5.2). An additional benefit of these architectures is that they are both fully-convolutional, which allows us to modify the video resolution at test time. We believe that the individual subnetworks can be improved to leverage the structure of our task more effectively, but we leave this for future work.

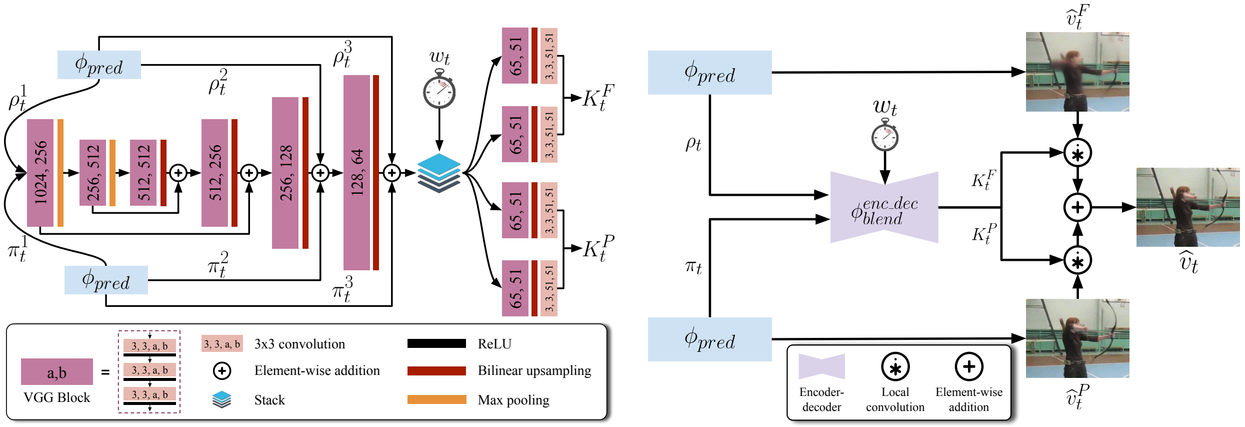
3.2.5.1 Bidirectional Video Prediction Network Details

MCnet Instead of learning the spatiotemporal representation via a single encoding network, MCnet [93] disentangles the spatial and temporal encoding via two sets of three VGG [94] encoder blocks each (the weights are not shared). The spatial encoder operates on the previous RGB video frame, and the temporal encoder operates on the difference between the last two video frames $v_{t-1} - v_{t-2}$. The temporal encoder also employs a Convolutional LSTM [95] to encode the temporal history of the sequence. After computing the spatial and temporal encodings, MCnet concatenates them along the feature dimension and decodes them into the next frame. The decoder also takes in the intermediate activations of corresponding VGG blocks in the content and temporal encoders, which are concatenated feature-wise and then fused via multiple convolution layers (residual layers).

Computing intermediate activations for TAI To obtain the intermediate features π_t and ρ_t to be fed to the TAI network, we concatenate the intermediate activations from corresponding VGG blocks in the spatial and temporal encoders. For example, π_t^1 is the concatenation of the activations from the first VGG blocks in the forward prediction, ρ_t^2 is the concatenation of the activations from the second VGG blocks in the backward prediction, and so on. We thus have three intermediate features for each of the forward and backward predictions for each time step, i.e. $\pi_t = \{\pi_t^1, \pi_t^2, \pi_t^3\}$ and $\rho_t = \{\rho_t^1, \rho_t^2, \rho_t^3\}$.

3.2.5.2 TAI Network Details

Similarly to the Separable Adaptive Kernel Network [92], our TAI network blends a pair of intermediate frames $(\widehat{v}_t^P, \widehat{v}_t^F)$ by first applying a unique, adaptive 2D kernel to each patch in the two input frames, and then summing the resulting images pixel-wise. The primary way in which our TAI network differs from the Separable Adaptive Kernel Network is in how the adaptive kernels are computed. Both use an encoder-decoder network structure [96] that outputs the adaptive kernels; however, the Separable Adaptive Kernel Network uses the two frames to interpolate as inputs to the encoder-decoder network, whereas we use intermediate activations from the Bidirectional Video Prediction Network, π_t and ρ_t, as well as the scaled time step $w_t = (t - p)/(m + 1)$. Note that we still apply the adaptive kernels to the intermediate frames $(\widehat{v}_t^P, \widehat{v}_t^F)$, not to the intermediate



(a) (b)

Figure 3.4: (a) The architecture of TAI's encoder-decoder network. (b) The TAI network applied to the intermediate predictions from the Bidirectional Video Prediction Network.

predictions π_t and ρ_t.

To be more precise, we take the scaled time step w_t and three sets of intermediate activations from the forward and backward predictions ($\pi_t = \{\pi_t^1, \pi_t^2, \pi_t^3\}$, $\rho_t = \{\rho_t^1, \rho_t^2, \rho_t^3\}$), and feed them to an encoder-decoder network to compute the parameters of the adaptive kernels K_t^P and K_t^F:

$$K_t^P, K_t^F = \phi_{blend}^{enc\_dec}\left(\pi_t, \rho_t, w_t\right) , \tag{3.8}$$

where K_t^P and K_t^F are 3D tensors whose height and width match the frame resolution and whose depth equals the number of parameters in each adaptive kernel. We inject the scaled time step by replicating it spatially and concatenating it to one of the decoder's hidden activations as an additional channel. As with the Separable Adaptive Kernel Network, we predict the parameters for a set of separable 2D kernels instead of standard 2D kernels to scale the size of the adaptive kernels more efficiently. The encoder-decoder network architecture is summarized in Figure 3.4a.

Afterwards, we apply the adaptive kernels to each input frame and sum the resulting images pixel-wise:

$$\widehat{v}_t(x,y) = K_t^P(x,y) * \mathcal{P}_t^P(x,y) + K_t^F(x,y) * \mathcal{P}_t^F(x,y) \,, \tag{3.9}$$

where $\widehat{v}_t(x,y)$ is the pixel value of the final prediction $\widehat{v}_t$ at (x,y), $K_t^{(\cdot)}(x,y)$ is the 2D kernel parameterized by the depth column of $K_t^{(\cdot)}$ at (x,y), $*$ is the convolution operator, and $\mathcal{P}_t^{(\cdot)}(x,y)$ is the patch centered at (x,y) in $\widehat{v}_t^{(\cdot)}$. We summarize our use of the TAI network in Figure 3.4b.

3.2.6 Training Strategy

To train bi-TAI, we use both reconstruction-based and adversarial objective functions, the latter of which has been shown by Mathieu et al. [97] to improve the sharpness of predictions. Elaborating on the adversarial objective, we train a discriminator D, which is a binary classification CNN, to distinguish between clips from the dataset and clips generated by bi-TAI. Meanwhile, we train bi-TAI—the "generator"—to not only fool the discriminator, but also generate predictions that resemble the ground truth.

We update the generator and the discriminator in an alternating fashion. In the generator update step, we update bi-TAI by minimizing the following structured loss:

$$\begin{aligned}\mathcal{L}_g = \alpha\mathcal{L}_{img}\left(\widehat{M}_V^P, M_V\right) + \alpha\mathcal{L}_{img}\left(\widehat{M}_V^F, M_V\right)\\ + \alpha\mathcal{L}_{img}\left(\widehat{M}_V, M_V\right) + \beta\mathcal{L}_{GAN}\left(\widehat{M}_V\right)\ , \end{aligned} \tag{3.10}$$

$$\mathcal{L}_{GAN}\left(\widehat{M}_V\right) = -\log D\left(\left[P_V, \widehat{M}_V, F_V\right]\right), \tag{3.11}$$

where α and β are hyperparameters to balance the contribution of the reconstruction-based loss $\mathcal{L}_{img}$ and the adversarial loss $\mathcal{L}_{GAN}$. Note that we supervise the final prediction $\widehat{M}_V$ as well as the intermediate predictions $\widehat{M}_V^P$ and $\widehat{M}_V^F$ simultaneously. The loss $\mathcal{L}_{img}$ consists of the squared-error loss $\mathcal{L}_2$ and the image gradient difference loss $\mathcal{L}_{gdl}$ [97], which encourages sharper predictions by penalizing the difference between the image gradients of the ground truth frames and the intermediate/final predictions at every pixel:

$$\mathcal{L}_{img}\left(\widehat{M}_V^{(\cdot)}, M_V\right) = \mathcal{L}_2\left(\widehat{M}_V^{(\cdot)}, M_V\right) + \mathcal{L}_{gdl}\left(\widehat{M}_V^{(\cdot)}, M_V\right)\ , \tag{3.12}$$

$$\mathcal{L}_2\left(\widehat{M}_V^{(\cdot)}, M_V\right) = \sum_t \left\| v_t - \widehat{v}_t^{(\cdot)} \right\|_2^2\ , \tag{3.13}$$

$$\mathcal{L}_{gdl}\left(\widehat{M}_V^{(\cdot)}, M_V\right) = \sum_{t,i,j,k} \left[|\nabla v_t - \nabla \widehat{v}_t^{(\cdot)}| \right]_{i,j,k}\ . \tag{3.14}$$

Here, $\widehat{M}_V^{(\cdot)}$ can be one of the intermediate predictions $\left\{\widehat{M}_V^P, \widehat{M}_V^F\right\}$ or the final prediction $\widehat{M}_V$. In the discriminator update step, we minimize the cross-entropy error:

$$\mathcal{L}_d = -\log D(V) - \log\left(1 - D\left(\left[P_V, \widehat{M}_V, F_V\right]\right)\right). \tag{3.15}$$

This loss encourages D to assign a high score to real video clips and a low score to clips that include generated middle frames. We use the same discriminator as Villegas et al. [93], but replace each layer that is followed by batch normalization [98] with a spectral normalization layer [99], which we have found results in more accurate predictions.

3.3 Experiments

3.3.1 Experimental Setup

3.3.1.1 Datasets

We perform our experiments on videos from several human action and object video datasets: KTH Actions [100], UCF-101 [101], HMDB-51 [102], and ImageNet-VID [103]. KTH Actions and UCF-101 are commonly used in video prediction and frame interpolation work [19], [93], [97], and HMDB-51 is a dataset we have added to further explore performance on challenging human action videos. ImageNet-VID is used to investigate performance on videos that do not primarily feature humans. We summarize the training and testing sets in Table 3.1, and now proceed to describe them in detail.

KTH Actions contains a total of 2,391 grayscale video clips with resolution 120×160 (height × width), which are divided into a standard training and testing set. Each video clip contains one of 25 subjects performing one of six actions (e.g. handwaving, jogging, boxing, etc.). We divide the standard training set into a smaller training set and a validation set based on the identity of the person in each video (the former includes subjects 1-14, and the latter includes subjects 15-16); these sets are used for training and hyperparameter search respectively. Following Villegas et al. [93], we reduce the resolution to 128×128. We train each model to predict up to five middle frames from up to five preceding and following frames (i.e. up to ten input frames in total); at inference time, we evaluate each model on its ability to predict either five or ten middle frames, given exactly five preceding and five following frames in both cases. The ten-frame case allows us to evaluate generalization performance (similarly to Villegas et al. [93], who double the number of frames to predict between training and testing time).

UCF-101 contains 13,320 RGB video clips from YouTube with resolution 240×320 across 101 action classes (e.g. horse riding, playing guitar, rowing, etc.). It provides three cross-validation folds for action recognition (each fold specifies a training and a test set); we take the test videos from the first fold as our test set and separate the remaining videos into our training and validation sets (clips are separated into an approximate 80-20 split such that the clips from any given source video do not appear in both sets). During training, we reduce the resolution of each video to 160×208 (due to the hardware limitations encountered with our bi-TAI model), and train each model to predict up to three middle frames from up to four preceding and following frames (i.e. up to eight input frames in total). At test time, we scale all videos to 240×320 resolution, and evaluate each model's ability to predict either three or five middle frames given exactly three preceding and three following frames.

Dataset	# source clips	Resolution (source)	Resolution (train)	Resolution (val/test)	Grayscale / color	Max p/f (train)	Max m (train)	p/f (val/test)	Small m (val/test)	Large m (val/test)
KTH [100]	2,391	120×160	128×128	128×128	Grayscale	5	5	5	5	10
UCF-101 [101]	13,320	240×320	160×208	240×320	Color	4	3	4	3	5
HMDB-51 [102]	6,849	240×var. (*)	160×208	240×320	Color	4	3	4	3	5
ImageNet-VID [103]	5,354	Varies	-	240×320	Color	-	-	4	3	5

Table 3.1: Summary of the training and testing sets used in our experiments. (*) The HMDB-51 source clips have varying aspect ratios, and thus varying widths.

Due to the large size of this dataset, we only evaluate on the first clip of each test video.

HMDB-51 contains 6,849 RGB video clips across 51 action classes (e.g. golf, hugging, somersaulting, etc.) from movie clips, YouTube, and other publicly available datasets; each video has a fixed height of 240 pixels. The dataset provides three cross-validation folds; we construct the training, validation, and test sets using the same strategy used for UCF-101. The rest of our experimental setup for HMDB-51 matches that of UCF-101.

ImageNet-VID is a video object detection dataset provided as part of the ILSVRC Challenge [103]. We use the 2015 version, which contains 5,354 RGB video clips across 30 animal and vehicle object classes. For this dataset, we use all models pre-trained on UCF-101 and evaluate on the provided test set. To preprocess the test set, we resize all videos to 240×320 and filter out videos with fewer than 13 frames.

Constructing video clips for training and testing During training, we construct minibatches by randomly sampling the number of preceding, middle, and following frames (p, m, and f respectively), selecting a video subclip with the appropriate number of frames, and then splitting that clip into the ground truth preceding, middle, and following sequence. Each video clip is randomly flipped horizontally or time-reversed before splitting with probability 0.5 for data augmentation.

We construct the validation and test sets differently for each of the three datasets. For KTH, we extract all subclips across all validation/test video clips from a sliding window of size $T = p+m+f$ and stride s. In our experiments, $p = f = 5$, m is 5 or 10, and s depends on the action class ($s = 3$ for the running and jogging classes, and $s = m$ for the walking, boxing, handclapping, and handwaving classes, following the stride selection process used by Villegas et al. [93]). For UCF-101, HMDB-51, and ImageNet-VID, we only evaluate each model on the first T frames of each video in the test set (following Villegas et al. [93]), where $T = p + m + f$, $p = f = 4$, and m is 3 or 5.

3.3.1.2 Baselines

As a sanity check, we compare our bi-TAI method to a linear time-weighted average of the last preceding frame and the first following frame, where the weights for the following and preceding frames for time t are $w_t = (t - p)/(m + 1)$ and $1 - w_t$ respectively. We refer to this baseline as `TW_P_F` for ***t***ime-***w***eighted ***p***receding and ***f***ollowing frame. We also compare bi-TAI to state-of-the-art methods for general frame inpainting, video prediction, and frame interpolation [19], [34], [93] to demonstrate that casting our video frame inpainting problem as a different video interpolation/extrapolation task does not yield optimal performance.

The first baseline, Newson et al. [34], is a general video inpainting method that iteratively fills in missing voxel patches with nearest neighbors in the unoccluded portion of the video, using a multi-resolution pyramid to improve the distance metric. For this method, we completely mask the middle frames and run the authors' publicly-available code to recover them. We omit their pre- and post-alignment of video frames in order to compare fairly against the other methods, which lack this step. Note that this method does not have a training phase (it has no parameters to tune based on training data).

The second baseline, MCnet [93], is a video prediction method that sequentially predicts frames by first decomposing the preceding clip into motion difference frames and an RGB content frame, and then regressing this representation to the next frame using an encoder-decoder network. We use this method to predict the middle frames given only the preceding frames as input (this method cannot take following frames because it is a video prediction method). We re-implement their code in PyTorch [104] based on their available implementation in TensorFlow [105]. We use the same loss functions as the original authors to train this model, but for a fairer comparison with bi-TAI, we improve their discriminator by using spectral normalization [99] instead of batch normalization [98].

The final baseline, Super SloMo [19], is a frame interpolation method that uses a CNN to predict the optical flow between the two input frames and the frame at an arbitrary intermediate time step. We use Super SloMo to predict each middle frame given only the last preceding frame and the first following frame as input (it cannot take multiple preceding or following frames because it is a frame interpolation method). Since the original code is unavailable, we re-implement it from scratch in PyTorch [104].

3.3.1.3 Training Hyperparameters

We train bi-TAI for 200,000 iterations with a batch size of 4. We use the Adam optimizer [106] with initial learning rate α = 1e-4, first decay rate $\beta_1 = 0.5$, and second decay rate $\beta_2 = 0.999$. In the generator loss, we set the weight of the reconstruction losses α to 1 and the weight of the adversarial loss β to 0.02. The discriminator's spectral normalization layers require a hyperparameter that specifies the number of power iterations used to approximate the spectral norm; we set this value to 3. We use Xavier initialization [107] for each convolutional layer and uniform initialization for each linear layer (with mean 0 and variance 1e-4 for the weights). The biases of each layer are initialized to 0.

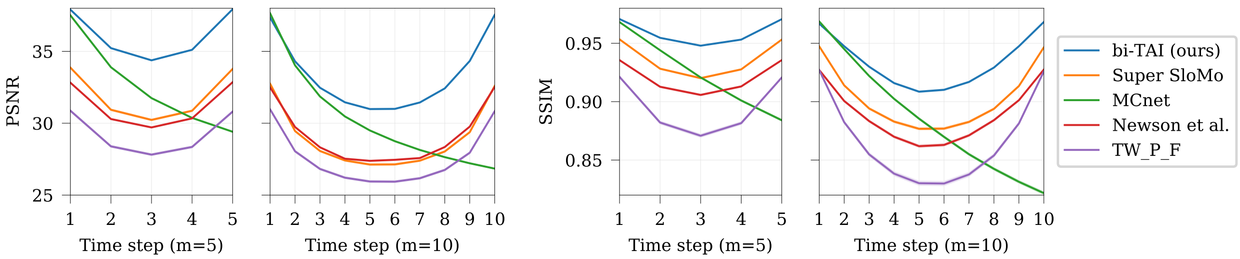


Figure 3.5: Performance on the KTH test set for each time step (higher is better).

	$m = 5$		$m = 10$	
Model	**PSNR**	**SSIM**	**PSNR**	**SSIM**
TW_P_F	29.25±0.053	0.8953±7.27e-4	27.56±0.051	0.8661±8.61e-4
Newson et al.	31.20±0.034	0.9205±4.57e-4	29.11±0.033	0.8890±5.94e-4
MCnet	32.58±0.032	0.9236±4.34e-4	30.21±0.032	0.8844±5.96e-4
Super SloMo	31.93±0.046	0.9365±4.13e-4	28.94±0.045	0.9028±5.78e-4
bi-TAI (ours)	***36.11±0.031***	***0.9594±2.52e-4***	***33.33±0.031***	***0.9340±3.73e-4***

Table 3.2: Performance on the KTH test set where each value is computed as the mean score across all predicted frames (higher is better). We stylize values that are higher than the others by a statistically significant margin.

For MCnet, we use the same optimization parameters and loss weights as the original authors, and the same number of spectral normalization power iterations as bi-TAI. For Super SloMo, the original authors use the Adam optimizer with an initial learning rate of 1e-4 and 500 total epochs, and divide the learning rate by 10 every 200 epochs. Since this model converges more rapidly on our datasets, we reduce the total epochs and the frequency of learning rate updates. Since the authors do not specify other Adam hyperparameters, we use the same ones used for bi-TAI.

The method by Newson et al. [34] requires hyperparameters for the size of the spatiotemporal patches and the number of levels in the multi-resolution pyramid. We set these values to (3, 3, 3) and 2 for KTH and (5, 3, 3) and 2 for UCF-101 and HMDB-51 (determined by performing grid search on the KTH and UCF-101 validation sets).

3.3.2 KTH

To evaluate the performance of our bi-TAI model and the proposed baselines, we report the Peak Signal-Noise Ratio (PSNR) and the Structural Similarity (SSIM) [44] between each predicted frame and the ground truth. We report these metrics to be consistent with existing video prediction literature [93], [97], but acknowledge that these metrics have a limited correlation with human perception as noted by Zhang et al. [43].

PSNR is defined as a logarithmic function of the inverse of pixel-wise mean-square error

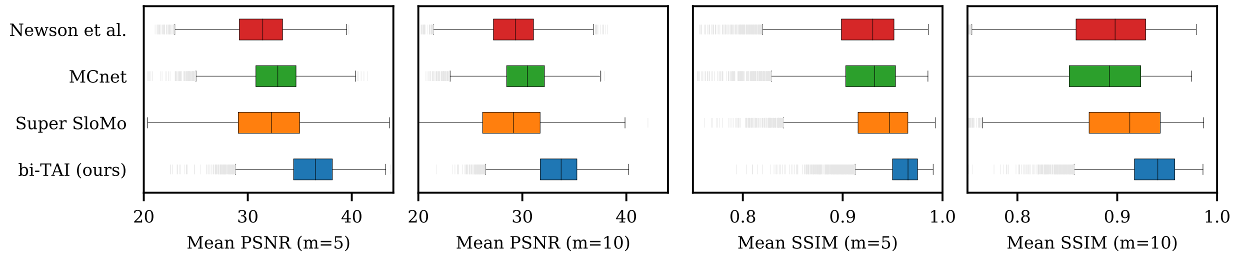


Figure 3.6: The distributions of performance on the video clips in the KTH test set. Performance per video is computed as the mean score across all predicted middle frames (higher is better). Outliers are shown as light gray lines.

between two images:

$$\mathrm{PSNR}(\widehat{x}, x) = 10 \log_{10} \left(\frac{255^2}{\mathrm{MSE}(\widehat{x}, x)} \right) \tag{3.16}$$

SSIM measures structural similarity as a function of means, variances, and covariances between many corresponding patches between two images, and its value can range between -1 and 1 (we refer the reader to the original paper [44] for more information). We use the implementations of PSNR and SSIM provided by scikit-image [108].

Figure 3.5 and Table 3.2 compare the quantitative performance of each method on the KTH test set when predicting five and ten middle frames. As expected, `TW_P_F` performs worse than our method and all state-of-the-art baselines because it does not use any temporal context or motion model to predict the middle frames. Newson et al. [34] does better, but its performance is restricted by its need to borrow spatio-temporal patches from the preceding and following sequence. MCnet yields good performance during the first few frames, but gradually does worse over time because it cannot reconcile the predicted middle frames with the following frames. Super SloMo yields the strongest performance across most metrics, but is restricted by only having access to one preceding and one following frame. Finally, our bi-TAI method significantly outperforms Super SloMo thanks to its ability to aggregate information across all preceding and following frames.

To understand the distribution of each model's performance across the dataset, we compute the average PSNR and SSIM score across all predicted frames for each video, and plot the distribution of these averages in Figure 3.6. Our method obtains the highest median and the smallest interquartile range, indicating that its predictions are more stable and of higher quality than the baselines.

Next, we demonstrate in Figure 3.7 that compared to the baselines, bi-TAI is better at predicting periodic motion, retaining body structure, and maintaining consistency with both the preceding and the following frames. Observe in the ground truth row that the man lowers his arms down to his sides and then raises them back up. MCnet and Newson et al. [34] fail to predict the arms

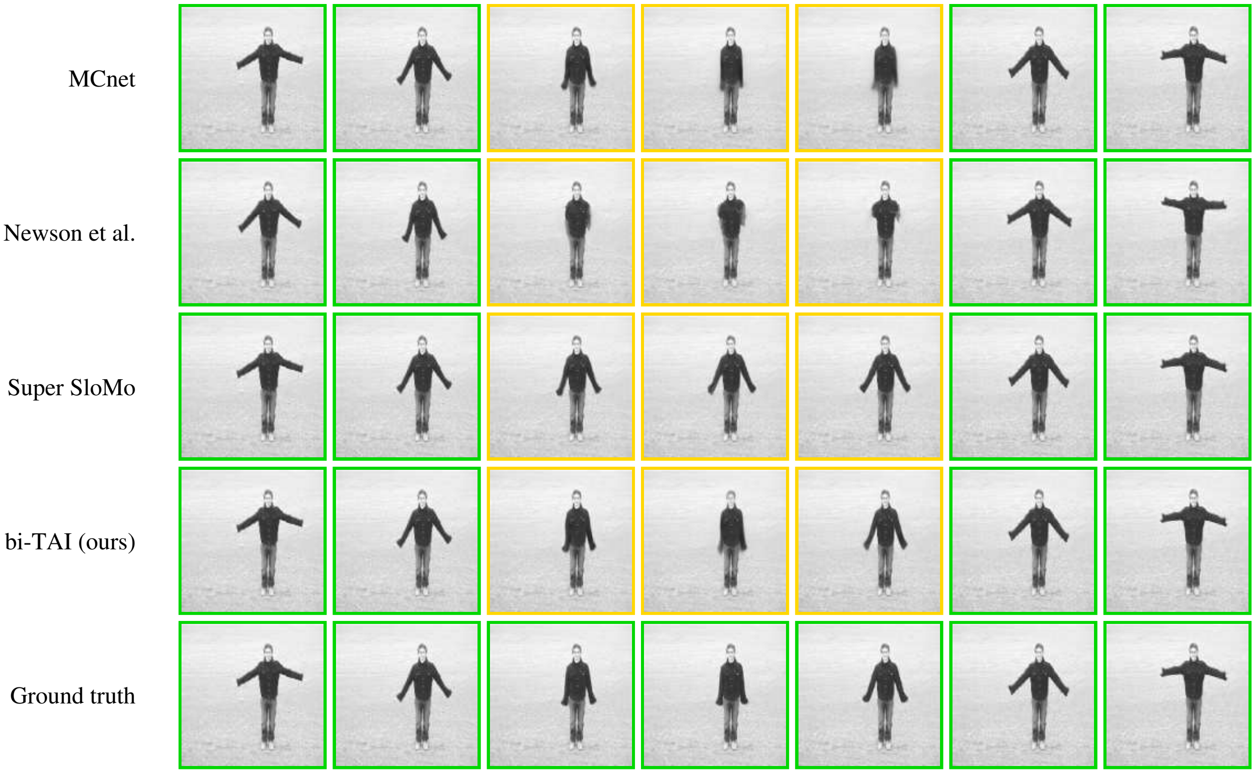


Figure 3.7: Qualitative results from the KTH dataset for predicting five middle frames from five preceding and five following frames (we depict every other frame for easier viewing). We indicate preceding and following frames with a green border, predicted middle frames with a yellow border, and ground-truth middle frames with a green border.

moving up, despite this motion being observable in the following frames (but recall that MCnet is a video prediction model, so it does not have access to the following frames). Meanwhile, Super SloMo fails to predict the arms moving all the way in, since it lacks the context from multiple preceding and following frames that indicates that the man is moving his arms rather than keeping them still. In contrast, bi-TAI predicts both the inward and the outward motion of the arms. We present additional results generated by our method in Figure 3.8.

In Figure 3.9, we present a negative result in which the strongest baseline, Super SloMo, outperforms bi-TAI quantitatively. In this example, Super SloMo predicts accurate, clear frames, whereas bi-TAI predicts slightly blurry frames (most evident in the most central middle frame). We have found that Super SloMo performs better than bi-TAI if it can accurately predict motion from two input frames (i.e. infer rate and direction of motion without the additional context provided by multiple preceding and following frames). However, the quantitatively lower performance shown in Figure 3.5 and Table 3.2 suggests that such cases are rare.

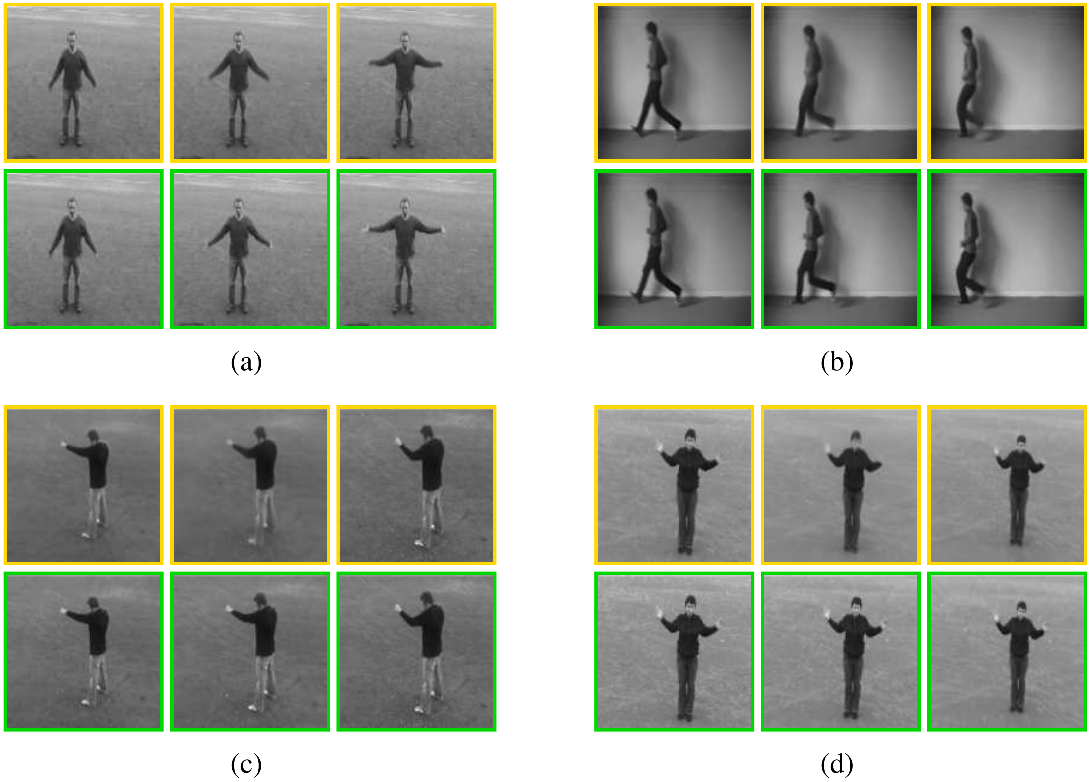

(a)
(b)
(c)
(d)

Figure 3.8: Additional predictions from bi-TAI on the KTH test set ($m = 5$). The yellow frames indicate predictions from bi-TAI, and green frames indicate the ground truth. We show the first, third, and fifth middle frame for each video.

3.3.3 Ablation Studies

In this section, we perform ablative studies to demonstrate that blending pairs of frames from the forward and backward predictions with a neural network—in particular, *one that is explicitly aware of the time steps corresponding to its inputs*—is key to producing high-quality predictions. For this experiment, we propose three approaches that perform bidirectional prediction as with our method, but blend corresponding pairs of intermediate frames in different ways:

- The *bidirectional simple average model (bi-SA)* blends a pair of frames by simply taking their average.

- The *bidirectional time-weighted average model (bi-TWA)* blends a pair of frames by taking a time-weighted average between them. The weights are $1 - w_t$ for the forward prediction frame and w_t for the backward prediction frame, where $w_t = (t - p)/(m + 1)$, p and m are the number of preceding and middle frames, and t is the index of a middle frame ($p < t \leq p + m$).

- The *bidirectional temporally-weighted interpolation model (bi-TWI)* is a variant of the bi-TAI model where the time weight w_t is used as a term for summing the modulated bidirectional

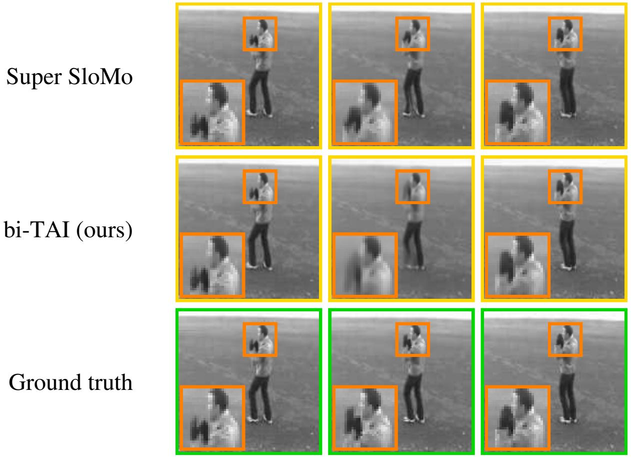


Figure 3.9: Negative result on the KTH test set ($m = 5$). We show the first, third, and fifth middle frames, and zoom in on the area indicated in orange.

predictions rather than as a feature channel in the encoder-decoder portion of the interpolation network ϕ_{blend}. To accomplish this, we first remove the injection of w_t as a feature channel within ϕ_{blend}. Then, we replace the simple sum in Eq. 3.9 with a time-weighted average. In short, we replace Eqs. 3.8 and 3.9 with Eqs. 3.17 and 3.18 respectively:

$$K_t^P, K_t^F = \phi_{blend}^{enc\_dec}\left(\pi_t, \rho_t\right) \ , \tag{3.17}$$

$$\begin{aligned} \widehat{v}_t(x,y) = (1 - w_t)\big[K_t^P(x,y) * \mathcal{P}_t^P(x,y)\big] \\ + w_t\big[K_t^F(x,y) * \mathcal{P}_t^F(x,y)\big] \ . \end{aligned} \tag{3.18}$$

As with bi-TAI, all of these models are trained from scratch.

In Fig 3.10b, we show a qualitative comparison between our approach and these methods when predicting the second-to-last middle frame (out of five). Unsurprisingly, bi-SA and bi-TWA produce ghosting artifacts (i.e. multiple faded copies of the actor appear in the predictions) because they do not possess a mechanism to make the bidirectional predictions consistent with each other. bi-TWI reduces the ghosting problem dramatically because the interpolation network can transfer information between the bidirectional predictions as it modulates them; however, ghosting artifacts do occasionally appear (e.g. we see an extraneous faded blob above the head). In contrast, bi-TAI manages to overcome the ghosting issue.

To understand what causes the difference in behavior between bi-TAI and its ablative variants, we visualize the intermediate pixel-space predictions from each model's Bidirectional Video Prediction Network and interpolation network. First, in Figure 3.10b, we compare the final predictions of each model to the corresponding outputs of the Bidirectional Video Prediction Network. We observe that the forward and backward predictions can differ substantially from each other (in terms of body

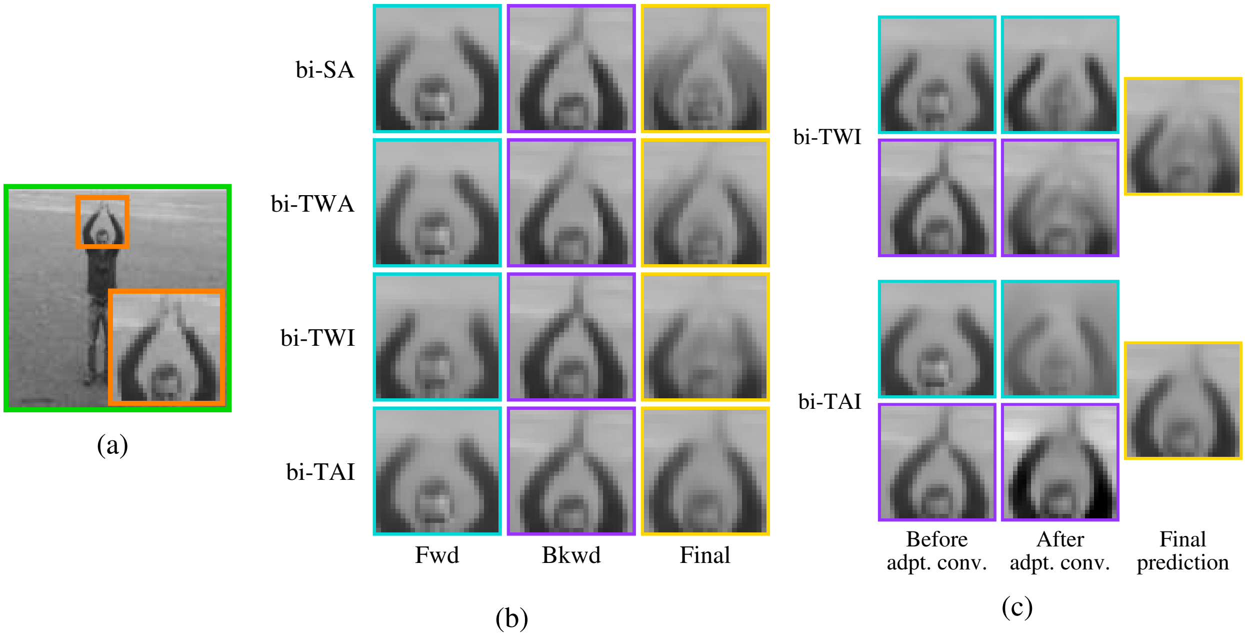


Figure 3.10: Qualitative comparison of the various intermediate predictions made by bi-TAI and its ablative variants (Sec. 3.3.3). We visualize the fourth predicted middle frame (out of five) from a “handwaving” video clip, and zoom in on a specific region (indicated in orange). (a) The ground-truth middle frame. (b) Comparison of the forward and backward predictions from the Bidirectional Video Prediction Network and the final predictions. (c) Inputs and outputs of the interpolation networks of bi-TWI and bi-TAI. The cyan images correspond to the forward prediction frame before and after adaptive convolution, and the purple images correspond to the backward prediction frame before and after adaptive convolution.

pose), but tend to be similar across methods. The discrepancy between the forward and backward predictions explains why simple and time-weighted averages lead to ghosting. bi-TWI and bi-TAI can reduce/eliminate ghosting by modulating the bidirectional predictions with convolutions before summing them.

Next, we emphasize the importance of making the interpolation network temporally-aware via injection of the time weight w_t. Figure 3.10c compares the predictions of bi-TWI and bi-TAI with respect to the three pixel-space representations that the interpolation network handles: (i) the two intermediate predictions from the Bidirectional Video Prediction Network; (ii) the two frames output by the interpolation network after adaptive convolution, but before time-weighted averaging (in bi-TWI) or simple summing (in bi-TAI); and (iii) the final predicted frame. Again, we show an instance of predicting the second-to-last frame, where the backward prediction has more weight than the forward prediction. Although the outputs of the Bidirectional Video Prediction Network are similar for both methods, bi-TWI distorts the man’s arms when it modulates the backward prediction. We believe this is because bi-TWI has no information about which of the two input frames is more reliable, which causes the inaccurate forward prediction to corrupt the backward

	$m = 5$		$m = 10$	
Model	**PSNR**	**SSIM**	**PSNR**	**SSIM**
bi-SA	33.69±0.031	0.9456±3.21e-4	30.95±0.030	0.9124±4.58e-4
bi-TWA	35.36±0.031	0.9553±2.73e-4	32.92±0.030	0.9296±3.94e-4
bi-TWI	***36.12±0.032***	0.9585±2.53e-4	***33.33±0.032***	0.9331±3.78e-4
bi-TAI (full)	***36.11±0.031***	***0.9594±2.52e-4***	***33.33±0.031***	***0.9340±3.73e-4***

Table 3.3: Performance of our bi-TAI model and the ablative variants described in Sec. 3.3.3 on the KTH test set. Each value is computed as the mean score across all predicted frames (higher is better). We stylize values that are higher than the others by a statistically significant margin.

prediction during the modulation process. When bi-TWI applies the time-weighted average to the two modulated outputs, the corruption heavily impacts the final prediction because the backward prediction has more weight. On the other hand, bi-TAI modulates the two predictions by enhancing the backward prediction and reducing the contribution of the forward prediction, which better matches our original intent of adding the interpolation network.

Moving on to quantitative results, Table 3.3 shows the average PSNR and SSIM score across all middle frames in the KTH test set when predicting five or ten middle frames with bi-TAI and the ablative methods. The quantitative results follow our qualitative analysis: bi-TWI and bi-TAI perform better than bi-SA and bi-TWA because they use an interpolation network to modulate the bidirectional predictions before summing. According to the quantitative metrics, it is not obvious whether bi-TAI or bi-TWI is better—bi-TAI performs better than bi-TWI according to SSIM, but comparably to bi-TWI according to PSNR. However, it is important to note that PSNR is less sensitive to structural changes in reconstructed images than SSIM (PSNR is a logarithmic function of *per-pixel* error, whereas SSIM is based on correlations between corresponding image *patches*). This suggests that SSIM is more suitable for evaluating predicted middle frames than PSNR, especially when comparing structural distortions that tend to occur with the ablated models, and further supports bi-TAI's superior performance.

3.3.4 UCF-101 and HMDB-51

We continue our analysis by comparing our model to the state-of-the-art baselines on video clips from the UCF-101 [101] and HMDB-51 [102] action classification datasets. Unlike the KTH dataset, these two datasets contain videos with unconstrained camera motion and dynamic lighting, making them substantially more challenging. Figure 3.11 shows the average PSNR/SSIM score for each time step when the number of middle frames is either 3 or 5 (recall that for these datasets, we train each method to predict at most three middle frames). Our model clearly outperforms the baseline methods when predicting the most central middle frames, but yields a less decisive improvement when predicting the first and last middle frame out of five. In Table 3.4, we report the performance of each model averaged across all predicted frames, which shows that our method

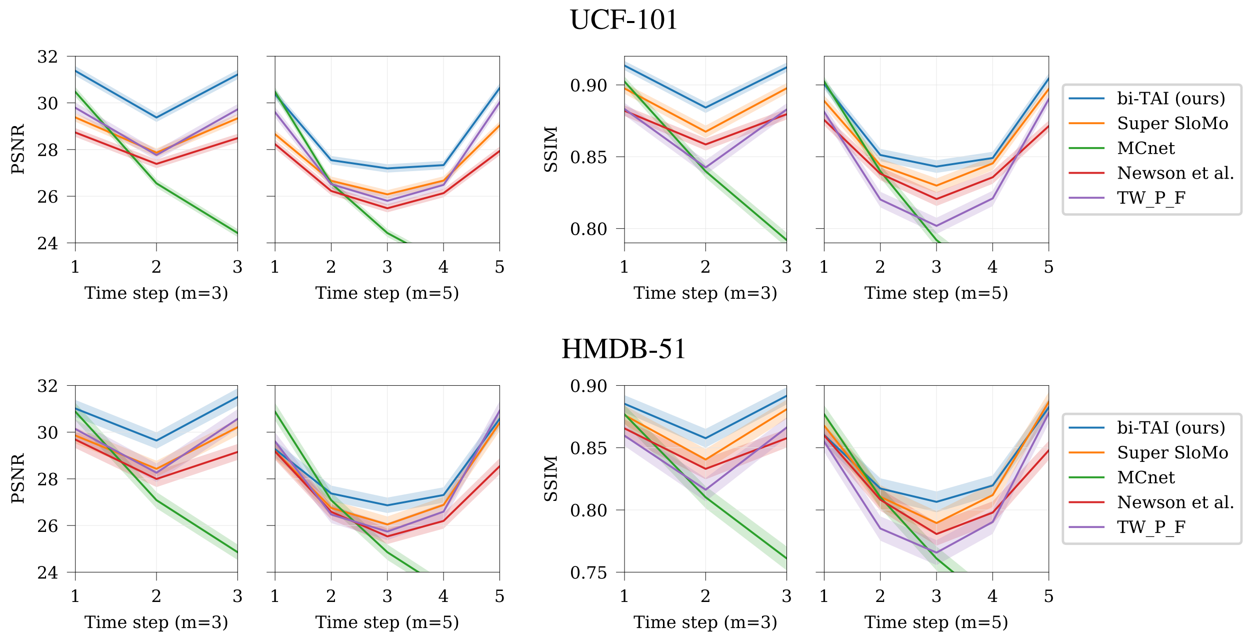


Figure 3.11: Performance on the UCF-101 and HMDB-51 test sets for each time step (higher is better). Light colors are used to highlight within two standard errors of each curve.

	UCF-101				HMDB-51			
	$m = 3$		$m = 5$		$m = 3$		$m = 5$	
Model	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**
TW_P_F	29.09±0.110	0.8696±2.203e-3	27.69±0.103	0.8429±2.371e-3	29.65±0.199	0.8474±4.047e-3	27.87±0.181	0.8148±4.155e-3
Newson et al.	28.20±0.091	0.8734±1.868e-3	26.80±0.087	0.8483±2.066e-3	28.94±0.168	0.8521±3.638e-3	27.21±0.156	0.8189±3.811e-3
MCnet	27.15±0.089	0.8447±2.117e-3	25.35±0.083	0.8067±2.308e-3	27.61±0.168	0.8160±4.055e-3	25.65±0.157	0.7725±4.311e-3
Super SloMo	28.86±0.088	0.8876±1.858e-3	27.42±0.087	0.8611±2.084e-3	29.50±0.162	0.8659±3.589e-3	27.85±0.153	***0.8333±3.810e-3***
bi-TAI (ours)	***30.65±0.095***	***0.9033±1.624e-3***	***28.62±0.091***	***0.8697±1.926e-3***	***30.72±0.175***	***0.8782±3.324e-3***	***28.28±0.165***	***0.8372±3.581e-3***

Table 3.4: Performance on the UCF-101 and HMDB-51 test sets where each value is computed as the mean score across all predicted frames (higher is better).

significantly outperforms the baseline methods except on HMDB-51 ($m = 5$) under SSIM, where it performs similarly to Super SloMo.

In Figure 3.12, we present two video clips from UCF-101 in which our method outperformed the most competitive baselines, Newson et al. [34] and Super SloMo [19]. We have found that bi-TAI is better at preserving object structure than the baselines in many cases, but it may also generate blurrier predictions. For example, in Figure 3.12a, Newson et al. [34] replaces parts of the torso with background pixels, and Super SloMo distorts the area around the arms and the back leg. Furthermore, the pose predicted by Super SloMo differs substantially from the ground truth. On the other hand, bi-TAI maintains a coherent structure for the athlete's body, and more accurately predicts his pose. We observe similar phenomena in Figure 3.12b: Newson et al. [34] replaces the person's feet with sidewalk pixels, Super SloMo produces distorted, inconsistent outlines of the front leg, and bi-TAI generates a more consistent, accurate body pose.

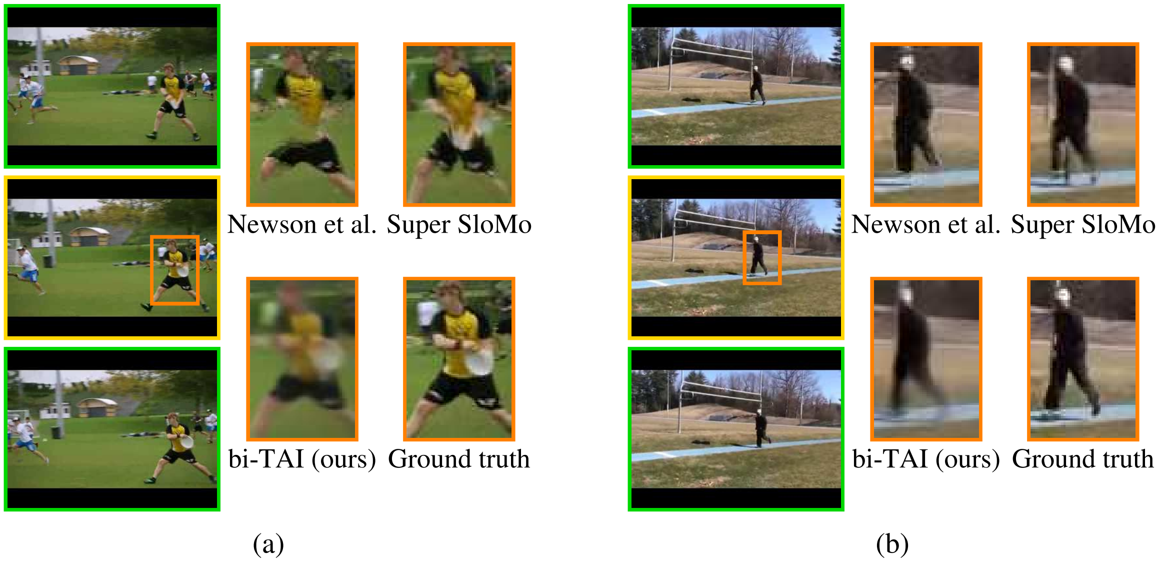


Figure 3.12: Qualitative results from the UCF-101 dataset ($m = 3$). On the left of each figure, we visualize the last preceding frame in green, the second middle frame in yellow, and the first following frame in green. On the right, we show the prediction from each method at the region indicated in orange.

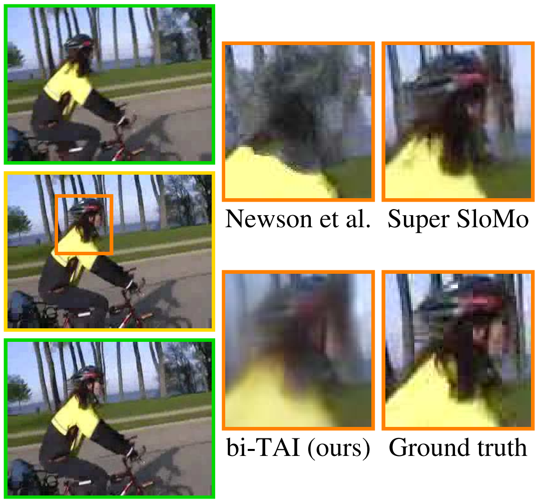
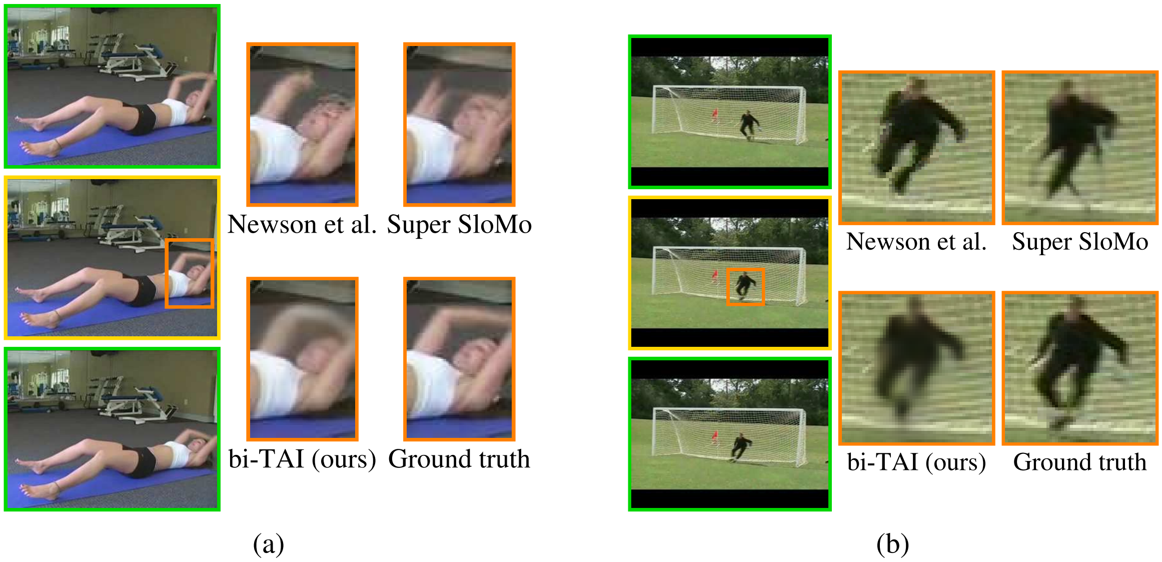


Figure 3.13: Failure case from UCF-101 (heavy camera motion).

In Figure 3.13, we show a negative result in which our bi-TAI method was outperformed by Newson et al. [34] and Super SloMo quantitatively. Our method performs worse when there is a large amount of camera motion: in these cases, the predictions become excessively blurry. Newson et al. [34] also performs poorly due to its inability to maintain the structure of objects; for example, in Figure 3.13, the face blends in with the trees. Super SloMo performs the best thanks to its ability to preserve structure and texture under heavy camera motion, but the camera poses in its predictions tend to differ from the ground truth (e.g. the biker's head is further to the left than in the ground



(a) (b)

Figure 3.14: Qualitative results from the HMDB-51 dataset ($m = 3$).

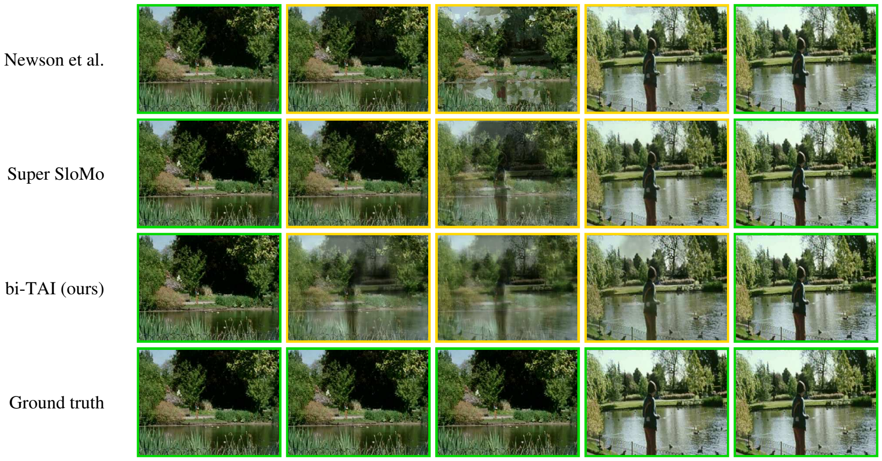


Figure 3.15: Failure case from HMDB-51 (shot transitions).

truth).

Moving on to the HMDB-51 dataset, we present qualitative results on sampled video clips in Fig 3.14. Again, we observe that our method is often able to preserve object structure more coherently and more accurately than the baselines. In Figure 3.14a, bi-TAI is the only method that successfully retains the entirety of the woman's right arm in its prediction. In Fig 3.14b, Super SloMo produces strange artifacts near the man's arms (likely remnants of the legs from the preceding and the following frames). Newson et al. [34] generates a more coherent body structure, but the pose is less accurate than in bi-TAI's prediction (e.g. the man's right leg does not bend downward). bi-TAI produces the most accurate and structurally coherent prediction among the evaluated methods.

In Figure 3.15, we present a video clip from HMDB-51 where Newson et al. [34] and Super SloMo outperformed our model. Compared to the baselines, our model is relatively worse at handling shot transitions because it tends to predict smooth motion. On the other hand, Newson et al. [34] and Super SloMo can generate abrupt transitions by borrowing pixels/patches solely from the appropriate input sequence. For example, in Figure 3.15, an outline of the boy appears in the first frame predicted by bi-TAI, whereas it does not appear in the first frames predicted by Newson et al. [34] and Super SloMo. Despite the existence of several shot transitions in HMDB-51 (where our model is at a disadvantage), bi-TAI still manages to achieve slightly higher performance overall; we suspect that the gap is even larger when only considering videos without shot transitions.

3.3.5 ImageNet-VID

We conclude our analysis with ImageNet-VID, where we pre-train each method on UCF-101 and evaluate on the ImageNet-VID test set. We investigate this setting to compare how well each method extrapolates motion from human action datasets to generic videos. In Figure 3.16, we show that our model achieves comparable or better quantitative performance than the baselines across all time steps. Table 3.5 reveals a similar trend—bi-TAI outperforms other models by a significant margin except under the SSIM metric when $m = 5$, where it performs on par with Newson et al. [34].

In Figure 3.17, we present cases in which our model captured motion better than the baselines. We observe in Figure 3.17a that the dog is most visible in bi-TAI's prediction. In Figure 3.17b, Newson et al. [34] erase the horse's legs, and Super SloMo produces ghosting leg artifacts. Our model accurately predicts the positions of the horse's legs, but produces a blurry result. The quantitative and qualitative results indicate that our method can extrapolate from human action videos to predict motion in general videos, although blur can still be observed as with UCF-101 and HMDB-51.

3.4 Conclusion

In this paper, we have tackled the video frame inpainting problem with bi-TAI, which generates two sets of intermediate predictions conditioned on the preceding and following frames respectively, and then blends them together with a novel TAI network. Our experiments on videos from multiple datasets show that our method generates smoother and more accurate predictions than state-of-the-art baselines, particularly on videos that contain articulated body motion and little camera movement. Furthermore, our in-depth analysis has revealed that our bi-TAI network successfully leverages time step information to reconcile inconsistencies in the intermediate predictions.

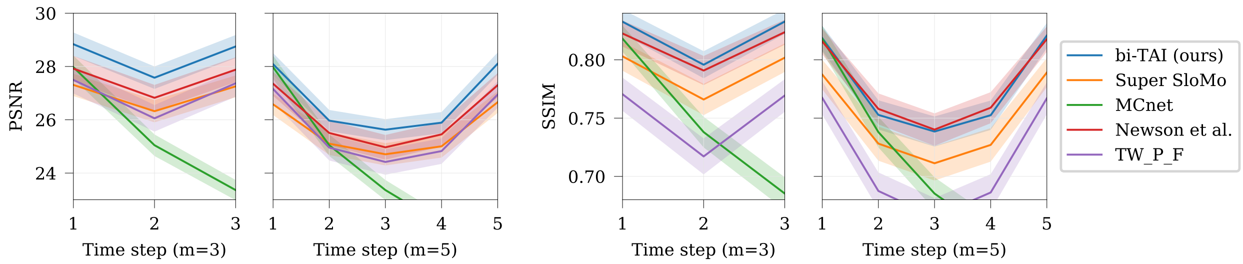
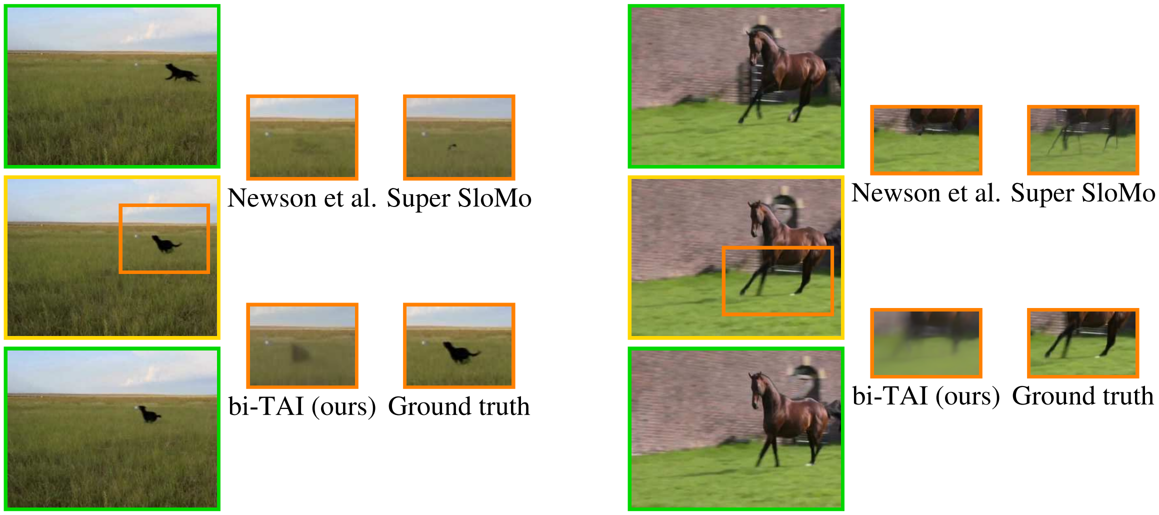


Figure 3.16: Performance on the ImageNet-VID test set for each time step (higher is better). Light colors are used to highlight within two standard deviations of each curve.

	$m = 3$		$m = 5$	
Model	**PSNR**	**SSIM**	**PSNR**	**SSIM**
TW_P_F	26.97±0.249	0.7523±7.11e-3	25.66±0.241	0.7147±7.35e-3
Newson et al.	27.54±0.236	0.8124±5.58e-3	26.12±0.226	***0.7781±6.03e-3***
MCnet	25.46±0.201	0.7471±6.22e-3	23.94±0.187	0.7033±6.44e-3
Super SloMo	26.96±0.205	0.7903±6.15e-3	25.61±0.203	0.7486±6.66e-3
bi-TAI (ours)	***28.39±0.213***	***0.8204±5.36e-3***	***26.74±0.200***	***0.7767±5.82e-3***

Table 3.5: Performance on the ImageNet-VID test set where each value is computed as the mean score across all predicted frames (higher is better). We stylize values that are higher than the others by a statistically significant margin.



(a)

(b)

Figure 3.17: Qualitative results from the ImageNet-VID dataset ($m = 3$).

There are several challenges that we plan to address in future work. First, compared to state-of-the-art baselines, our model sometimes generates relatively blurry results, especially under heavy camera motion. This is a common artifact of "pixel synthesis" methods that generate pixels from scratch. Taking inspiration from optical flow-based approaches, which borrow pixels from the input, may help alleviate the blurriness problem. Additionally, our network is very large because it encodes and decodes several pixel-level representations of the preceding, middle, and following frames; this places a limit on the sizes of video clips that can be generated or used to train our model. To address this problem, we aim to explore more tightly-integrated network architectures that generate only one pixel-level prediction. Finally, we plan to explore methods that exploit semantic knowledge about the video content, e.g. by modeling human poses or the periodicity of certain actions.

CHAPTER 4

Temporally-Consistent Video-to-Video Translation

4.1 Introduction

Developments in both large-scale datasets and deep neural networks (DNNs) have led to incredible advancements in image-to-image [110], [111] and video-to-video [31], [112]–[114] translation tasks such as color restoration [115], [116], super-resolution [117], inpainting [57], [109], [118], and style transfer [86], [87]. But compared to images, videos pose an additional challenge: not only do the video frames need to satisfy the intended translation, they must also be temporally consistent. Otherwise, they will exhibit flickering artifacts.

Existing video-to-video translation techniques address temporal consistency through one of two types of strategies. The first incorporates temporal consistency directly into the method through optical flow-based losses [20], [21], [27], [36], [56] or network layers that operate on the time dimension, e.g., 3D convolutional layers [37] or recurrent layers [56]. These techniques leverage self-supervised video data and tailor models to specific tasks using relevant losses, e.g., reconstruction error [56], [57], [109] or style loss [21], [86]. However, they require models and losses that are defined exclusively on videos and tuned for a specific application.

The other type of strategy uses a *blind video consistency* model to reduce flicker in a frame-wise translated video as a post-processing step [42], [119]–[121]. For example, Lai et al. [42] train a recurrent encoder-decoder network to improve the temporal consistency of independently processed frames via warping and structural preservation losses. Blind video consistency methods relax the need for task-specific video models and losses, and also enables image-to-image models to be applied immediately to videos without sacrificing consistency. However, they require dense correspondences between unprocessed frames during optimization, making them unsuitable for tasks in which certain input regions have no meaningful structure, e.g., video inpainting.

Similarly to blind video consistency methods, we opt to impart image-to-image models with temporal consistency, motivated primarily by generalization issues inherent to video-tailored approaches. To elaborate, consider in Figure 4.1 a failure case from the otherwise impressive state-of-the-art video inpainting network VINet [56]; here, it fails to hallucinate a sensible texture for the missing

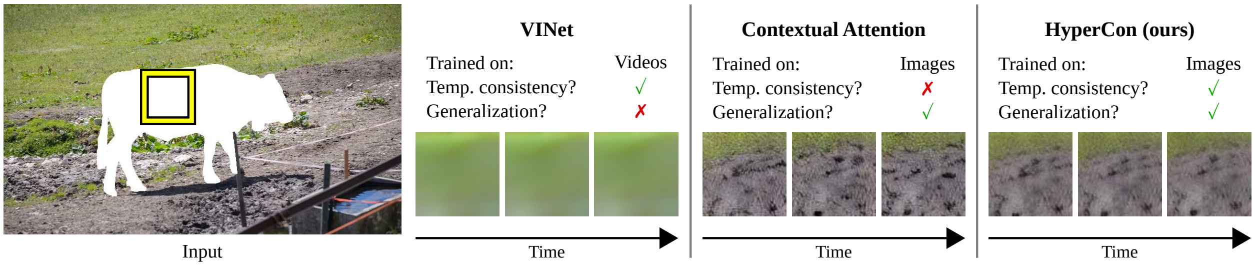


Figure 4.1: Video-to-video translation models designed and trained from scratch, e.g., VINet [56] are temporally consistent, but exhibit poor generalization performance due to the limited size of high-fidelity video datasets (note the lack of defined texture). Image-to-image models, e.g., Contextual Attention [109], generalize well thanks to large image datasets, but lack temporal consistency (note the changing texture). HyperCon leverages the generalization performance conferred by image datasets while enforcing the temporal consistency properties of video-to-video models.

region. Now consider a state-of-the-art *image* inpainting model, Contextual Attention [109]: this method produces realistic textures on the same example (but exhibits temporal inconsistency). The reason for this difference lies in the diversity of data used to train each model. Whereas the video model was trained on about 5,000 examples from one of the largest video segmentation datasets to date [51], the image model was trained on over 1,000,000 images [54]. Regardless of application, the vast scale of image datasets enables image models to encapsulate broader visual knowledge than video models trained from scratch and, as a result, better generalize to new data. Because storage and cost limitations make it intractable to collect high-quality video datasets as diverse as modern image datasets, video-tailored models are doomed to generalize poorly compared to image models.

To overcome this challenging generalization issue, we propose a method of *image-to-video model transfer for video-to-video translation tasks*. Rather than optimize an image or video model from scratch, we aim to transform a black-box image-to-image translation model into a strong video-to-video translation model without fine-tuning—specifically, to automatically induce temporal consistency while achieving the same visual effect as the image-to-image model. Compared to prior techniques in blind video consistency [42], [119], which target these goals and therefore fall under the same problem space, ours broadens the scope of applicable tasks to include those in which the input and output videos have differing visual structure, such as video inpainting and edge-deforming style transfer (e.g., the *mosaic* style from PyTorch's examples repository).

Key to our approach is the view of image-to-image models as noisy models in which small changes in the input lead to substantial changes in the output (e.g., Figure 4.1, middle). To strengthen the desired signal and filter out noise, we synthesize several perturbed, but related predictions and aggregate over them. We obtain perturbations in a way that conditions on multiple neighboring frames to enhance temporal consistency. Our *hyperconsistency* approach (*HyperCon* for short) implements these principles by inserting frames into the video with a frame interpolation network,

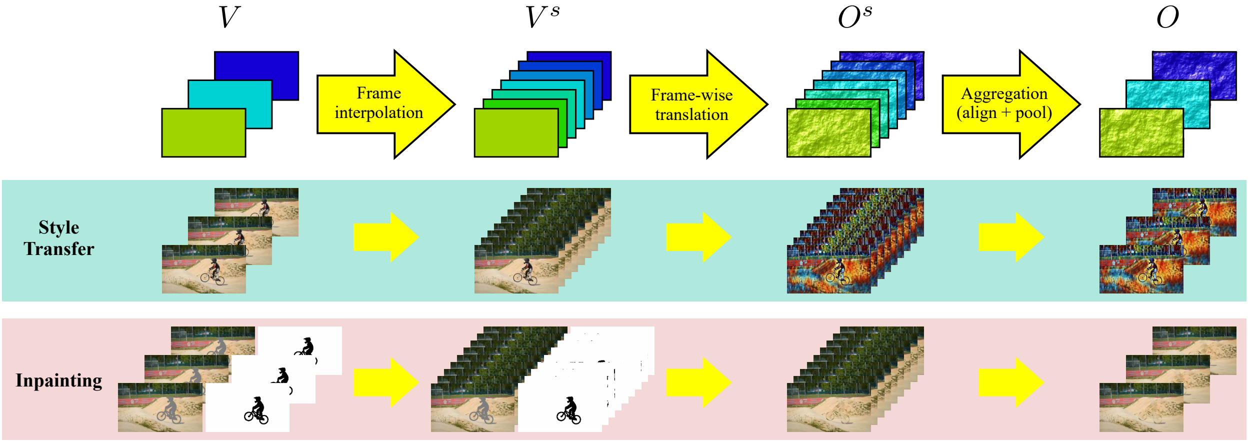


Figure 4.2: Visual overview of our Hyperconsistency (HyperCon) method. We begin by artificially inserting frames into the input video V with a frame interpolation network to produce an interpolated video V^s. Then, we independently translate each frame in the interpolated video with an image-to-image translation model. Finally, we aggregate frames (i.e., align with optical flow and pool pixel-wise) within a local sliding window to produce the final temporally consistent output video O. This can be applied to tasks with or without masked inputs (e.g., inpainting and style transfer, respectively).

translating the interpolated video's frames independently, and aggregating within overlapping windows of appropriate stride to obtain a final video whose length matches the original (Figure 4.2).[1]

As the first method among image-to-video model transfer techniques to reason in interpolated video space, HyperCon forgoes the traditional post-processing paradigm by integrating frame-wise translation *within* itself as an intermediate step, not as a step that precedes it. Since it does not require dense correspondences in the unprocessed input video, it performs well on video-to-video translation tasks with or without masked inputs—e.g., inpainting and style transfer respectively—as verified by our extensive experiments across these two widely differing applications. HyperCon outperforms a prior state-of-the-art video consistency model [42] in terms of reducing flicker and adhering to the intended translation; when combined with a strong image inpainting method, it also produces better predictions than a state-of-the-art video inpainting model [56]. It achieves competitive performance in both tasks *despite not being trained with any masked or stylized videos*.

Our contributions are as follows. First, we motivate image-to-video model transfer as a way to leverage the superior generalization performance of image models for video-to-video translation without sacrificing temporal consistency. Second, we propose HyperCon, which supports a wider span of tasks than prior video consistency work thanks to its support for both masked *and* unmasked inputs. Finally, we show that HyperCon performs favorably compared to state-of-the-art video consistency and inpainting methods without the need to be fine-tuned on these tasks. Our project

[1] HyperCon was originally proposed by this dissertation's author in [122].

website is available at ryanszeto.com/projects/hypercon.

4.2 Related Work

Image-to-image translation [110], [111], [123]–[126] has garnered significant attention thanks to advancements in conditional generative adversarial networks [127]. This movement has also helped spur interest in video-to-video translation, particularly in two broad cases: (i) the paired setting [31], [128], which learns a one-way mapping between a source and a target modality given corresponding video pairs, and (ii) the unpaired setting [112]–[114], which learns a bidirectional mapping between two modalities given two sets of videos representing them. For specific tasks, performance can be further improved by incorporating a task-based objective, e.g., a pixel-wise reconstruction loss for inpainting [56], [57], [109], [118], [129], [130] or style reconstruction and total variation losses for style transfer [21], [86]. Unlike these works, our goal is *not* to train an image or video model optimally from scratch, but to transform a trained image model into a strong, temporally consistent video model, which boosts generalization performance as motivated by Figure 4.1.

Among existing work, our goals for image-to-video model transfer best align with those of blind video consistency [42], [119]–[121], which reduces flicker from frame-wise translated videos by leveraging correspondences in the input. For example, Bonneel et al. [119] minimize an energy functional with a flow-based temporal consistency term and an edge-based scene consistency term, both of which are defined between corresponding frame-wise translated and output frames. Lai et al. [42] extend their work by training a DNN with loss terms that emulate their functional. These methods require accurate dense correspondences between input frames during optimization, making them unsuitable for video-to-video translation tasks where certain regions have no meaningful structure, e.g., inpainting. Our method does not rely on dense correspondences in the input, allowing it to be applied to such tasks. Additionally, our method forgoes the typical post-processing paradigm of prior work [119], [120] by integrating frame-wise translation *within* itself as an intermediate step rather than as a step that precedes it.

4.3 HyperCon

Given an input video $V = \{v_1, \ldots, v_N\}$, our goal is to generate an output video $O = \{o_1, \ldots, o_N\}$ representing the N frames of V translated by some image-to-image model g. The frames of O should closely resemble the frames of V translated frame-wise by g; at the same time, O should be temporally consistent, i.e., exhibit as few flickering effects as possible. We quantify these notions of resemblance and consistency concretely in Sections 4.4.2 and 4.5.2.

With HyperCon (Figure 4.2), we generate O as follows. First, we artificially insert i frames between each pair of frames in V with a frame interpolation network (Section 4.3.1). Denoting this interpolated version of V as $V^s = \{v^s_1, \ldots, v^s_{N'}\}$, where N' is the number of frames in the interpolated video, we then independently translate frames in V^s with g, yielding $O^s = \{o^s_1, \ldots, o^s_{N'}\}$ (Section 4.3.2). Finally, we aggregate frames in O^s over a temporal sliding window with appropriate stride to produce the frames of the final output video O (Section 4.3.3). Additional considerations for masked inputs are discussed in Section 4.3.4.

4.3.1 Generating the Interpolated Video

To generate the interpolated video V^s, we insert i interpolated frames between each pair of frames in V, which essentially allows us to obtain several perturbed versions of each input frame for translation. We opt for a vector-based sampling method for frame interpolation instead of a kernel-based one[2] which, as we justify in Section 4.3.4, allows us to handle the case of masked inputs appropriately. Specifically, for each pair of consecutive frames v_a and v_{a+1} ($a \in 1, \ldots, N-1$) and intermediate frame index $b \in \{1, \ldots, i\}$, we predict two warping grids $(F^s_{a+b' \to a}, F^s_{a+b' \to a+1})$ and a weight map $w_{a+b'}$ (where $b' \equiv \frac{b+1}{i+1}$) with some function wrpgrd (e.g., a pre-trained DNN), and use them to generate the corresponding interpolated frame v^s_j ($j \in \{1, \ldots, N'\}$):

$$(F^s_{a+b' \to a}, F^s_{a+b' \to a+1}, w_{a+b'}) = \mathrm{wrpgrd}(v_a, v_{a+1}, b')\,, \tag{4.1}$$

$$\begin{aligned} v^s_j = {} & (1 - w_{a+b'}) \odot \mathrm{warp}(v_a, F^s_{a+b' \to a}) \\ & + w_{a+b'} \odot \mathrm{warp}(v_{a+1}, F^s_{a+b' \to a+1})\,. \end{aligned} \tag{4.2}$$

$\odot$ is an element-wise product; warp(v, F) bilinearly samples from v via displacements specified by vector field F.

4.3.2 Translating the Interpolated Video

At this point, we have computed the interpolated video V^s. We generate the translated interpolated video O^s by simply translating each frame in V^s independently:

$$o^s_j = g(v^s_j), \quad j \in \{1, \ldots, N'\}\,. \tag{4.3}$$

Clearly, O^s is not temporally consistent. However, we expect that most spatial regions in this video will exhibit consensus within small temporal windows. For example, a patch might have a distinct color profile in one frame, but a common color profile in the other frames in the local temporal window. Since we have more frames in O^s than frames needed in the output, we can remove the

[2] This distinction is discussed in more detail in Reda et al. [131].

spurious artifacts of frame-wise translation by mapping several neighboring frames in O^s to one frame in our desired output video O. We call this mapping *temporal aggregation* (Section 4.3.3).

4.3.3 Temporal Aggregation

We perform temporal aggregation over a sliding window on the translated interpolated video O^s (Figure 4.3). The stride is such that the frame in each window's center, i.e., the reference frame, corresponds to a frame from the (non-interpolated) input video V, resulting in N windows. Within each window, we align the off-center frames, i.e., the context frames, to the reference frame via optical flow warping, and then pool the reference and aligned context frames pixel-wise (e.g., with a mean or median filter) to produce a final frame in the output video O. Note that we compute the flow between *interpolated, translated* frames O^s, *not* the unprocessed input frames V like prior work [119].

More precisely, for an interpolated frame index $j \in \{1, 1+(i+1), \dots, N'-(i+1), N'\}$, we first estimate the optical flow between reference frame o_j and each context frame in $\{o^s_{j-d\gamma}, o^s_{j-d(\gamma-1)}, \dots, o^s_{j-d}, o^s_{j+d}, \dots, o^s_{j+d(\gamma-1)}, o^s_{j+d\gamma}\}$ (denoted $F^a_{j\to(*)}$), where γ and d respectively parameterize the number of frames in the sliding window and a temporal dilation factor. We then warp the context frames to align them to o^s_j, and afterwards perform pixel-wise pooling over o^s_j and the warped context frames:

$$o'_{j,k} = \begin{cases} o^s_j & k = 0 \\ \text{warp}(o^s_{j+dk}, F^a_{j\to j+dk}) & k \neq 0 \end{cases}, k \in \{-\gamma, \dots, \gamma\}, \tag{4.4}$$

$$o_j = \text{pool}\big(o'_{j,k} \mid k \in \{-\gamma, \dots, \gamma\}\big) . \tag{4.5}$$

Pooling is applied over values per spatial location, color channel, and time step; i.e., if $P = \text{pool}(I_1, I_2, \dots)$, then

$$P(l_h, l_w, l_c) = f\big(I_1(l_h, l_w, l_c), I_2(l_h, l_w, l_c), \dots\big) , \tag{4.6}$$

where $P(l_h, l_w, l_c)$ and $I(l_h, l_w, l_c)$ denote the value of 3D image tensors P and I at location (l_h, l_w, l_c), and f is a mean or median operation. In the cases where the sliding window samples outside the valid frame range, we only align and pool over valid frames.

To illustrate why HyperCon induces temporal consistency, we visualize intermediate outputs for style transfer in Figure 4.3. Even among interpolated frames with similar appearances, flickering artifacts can occur. By selecting pixel values by a majority vote over several interpolated frames, our method automatically incorporates stable components into the final prediction, thereby reducing flicker.

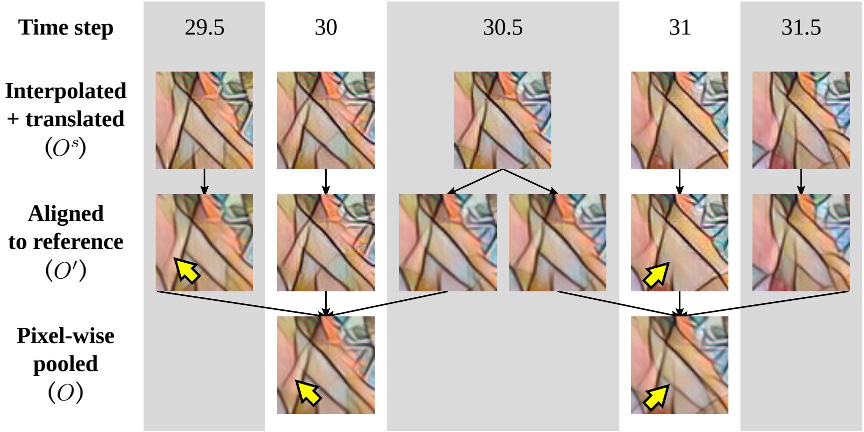

Time step
29.5
30
30.5
31
31.5
Interpolated + translated (O^s)
Aligned to reference (O')
Pixel-wise pooled (O)

Figure 4.3: Temporal aggregation. Context frames from the translated interpolated video O^s are aligned via optical flow to reference frames associated with integer time steps (white columns) and then pooled at each pixel location to generate the final video O. Despite inconsistencies between aligned frames (e.g., near the arrows), temporal aggregation selects stable components by majority vote.

4.3.4 HyperCon for Masked Videos

Having described HyperCon in the unmasked input case in Sections 4.3.1-4.3.3, we now extend it to handle tasks in which the input frames have masked pixels (e.g., inpainting). This case differs from the unmasked input case in three ways. First, in addition to the normal RGB video V, we now have as input a mask video $M = \{m_1, \ldots, m_N\}$ in which 1 marks an unmasked pixel and 0 marks a masked pixel. Second, when generating the interpolated data, we must create an interpolated mask video $M^s = \{m^s_1, \ldots, m^s_{N'}\}$ to accompany the interpolated RGB video V^s. Finally, the image-to-image translation model g now takes a mask as input in addition to an RGB video frame.

We modify the interpolated video generation step (Section 4.3.1) to produce both V^s and M^s; this is done by generating i interpolated frames between each pair of frames in V and M. For this to be valid, the motion of the interpolated mask video must match that of the interpolated RGB video—for example, if we interpolate the motion of a removed person, the mask must cover that person throughout the interpolated sequence. If this is not handled properly, we risk polluting the final result with mask placeholder values. Thus, we opt for a vector-based sampling method for frame interpolation instead of a kernel-based one, since the same warping grid can be applied to both RGB and mask frames to achieve the desired result.

To generate V^s, recall that we predict warping grids and weight map $(F^s_{a+b' \to a}, F^s_{a+b' \to a+1}, w_{a+b'})$ from frames in V using Equation 4.1. To obtain the interpolated masks M^s, we apply these parameters to the masks in M and follow up with a thresholding operation:

$$\begin{aligned} \dot{m}_j^s &= (1 - w_{a+b'}) \odot \text{warp}(m_a, F^s_{a+b' \to a}) \\ &\quad + w_{a+b'} \odot \text{warp}(m_{a+1}, F^s_{a+b' \to a+1}) \,, \qquad (4.7) \\ m_j^s &= \text{thresh}(\dot{m}_j^s, 1) \,. \qquad (4.8) \end{aligned}$$

Warping the masks in this way allows us to detect the "partially-masked" pixels in v_j^s, i.e., the ones that received a contribution from a masked pixel in either v_a or v_{a+1}. Specifically, if a pixel in $\dot{m}_j^s$ is not 1, then the warping operation used a source value of 0 from m_a or m_{a+1}, which corresponds to borrowing from a masked pixel. Thus, thresholding turns partially-masked pixels into fully-masked pixels in the interpolated masks so that the subsequent translation step is not incorrectly influenced by these pixels.

At this point, we have generated the interpolated RGB and mask videos V^s and M^s. We apply the image-to-image model g to them:

$$o_j^s = g(v_j^s, m_j^s), \quad j \in 1, \dots, N' \,, \qquad (4.9)$$

and then apply temporal aggregation (Section 4.3.3) as usual.

4.3.5 HyperCon Implementation Details

For the frame interpolation step, we use Super SloMo [19] as our wrpgrd function to predict warping grids and weight masks. This method is well-suited for our approach since it predicts warping parameters for multiple intermediate time steps, contrasting with kernel-based sampling methods that interpolate one frame [92]. To estimate optical flow in the temporal aggregation step, we use a third-party implementation of PWC-Net from Sun et al. [90]. Since HyperCon is agnostic to the specific instantiations of frame interpolation and flow estimation, we have chosen state-of-the-art models to demonstrate the full potential of our overall approach. Although we do not fine-tune network weights, we observe good performance regardless and postulate that performance can be further improved by fine-tuning.

4.4 Experiments: Video Style Transfer

To demonstrate HyperCon on unmasked inputs, we apply it to video style transfer. For the frame-wise style transfer subroutine, we use the Fast Style Transfer (FST) models from Johnson et al. [86] with the pre-trained *mosaic* and *rain-princess* weights from PyTorch's examples repository.

4.4.1 Datasets

For evaluation, we use the ActivityNet [132] and DAVIS [45] video datasets, which primarily consist of dynamic indoor and outdoor scenes of animals and people. We split them into validation and test sets to validate HyperCon hyperparameters and evaluate performance, respectively. For ActivityNet, we manually curate clips from the official training/validation and test splits (~100 videos per split with ~100 frames per video), filtering by properties of high-quality videos such as non-negligible motion, no splash screens, one continuous camera shot, etc. For DAVIS, we use the official training/validation set for validation, and combine the development and challenge sets from the DAVIS 2017 and 2019 challenges to produce two test sets, one for each year. We pre-process all videos by resizing and center-cropping them to 832×480 resolution, and scaling RGB values to (-1, 1).

4.4.2 Evaluation Metrics

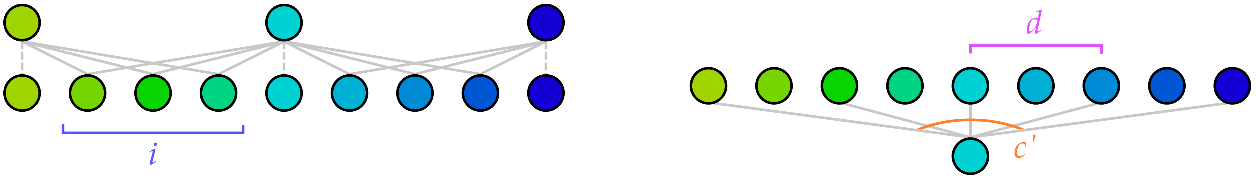
For video style transfer, our goal is to transfer the desired style to all video frames while minimizing flickering effects (i.e., maximizing temporal consistency) in the output video as much as possible. To evaluate temporal consistency, we measure warping error E_{warp} [42] and patch-based consistency measures C_{PSNR} and C_{SSIM} [21]. E_{warp} is defined as the mean of e_{warp} over all consecutive pairs of output frames (o_a, o_{a+1}), where

$$e_{\text{warp}}(o_a, o_{a+1}) = \frac{1}{\sum_p M_a^f(p)} \sum_p M_a^f(p) \| D_a(p) \|_2^2 \,. \tag{4.10}$$

Here, $D_a = o_a - \text{warp}(o_{a+1}, F_{a \to a+1})$ (where $F_{a \to a+1}$ is the estimated flow between input frames v_a and v_{a+1}); p indexes the pixels in the frame; and M_a^f is a mask that indicates pixels with reliable flow (1 for reliable, 0 for unreliable). M_a^f is computed based on flow consistency and motion boundaries as defined by Ruder et al. [20]. C_{PSNR} is defined as the mean of c_{PSNR} over all consecutive pairs of output frames, where c_{PSNR} is computed by taking a random 50×50 patch in frame o_a and computing the maximum PSNR between it and all patches in its spatial neighborhood in the next frame o_{a+1} (within a Chebyshev distance of 20 pixels). C_{SSIM} is defined similarly to C_{PSNR}, except it computes Structural Similarity [44] in place of PSNR. To quantify how well a method adheres to the intended style, we measure Frechét Inception Distance (FID) [41] between the set of all frames generated by frame-wise translation and the set of all frames generated by the method under evaluation.

4.4.3 Hyperparameter Analysis

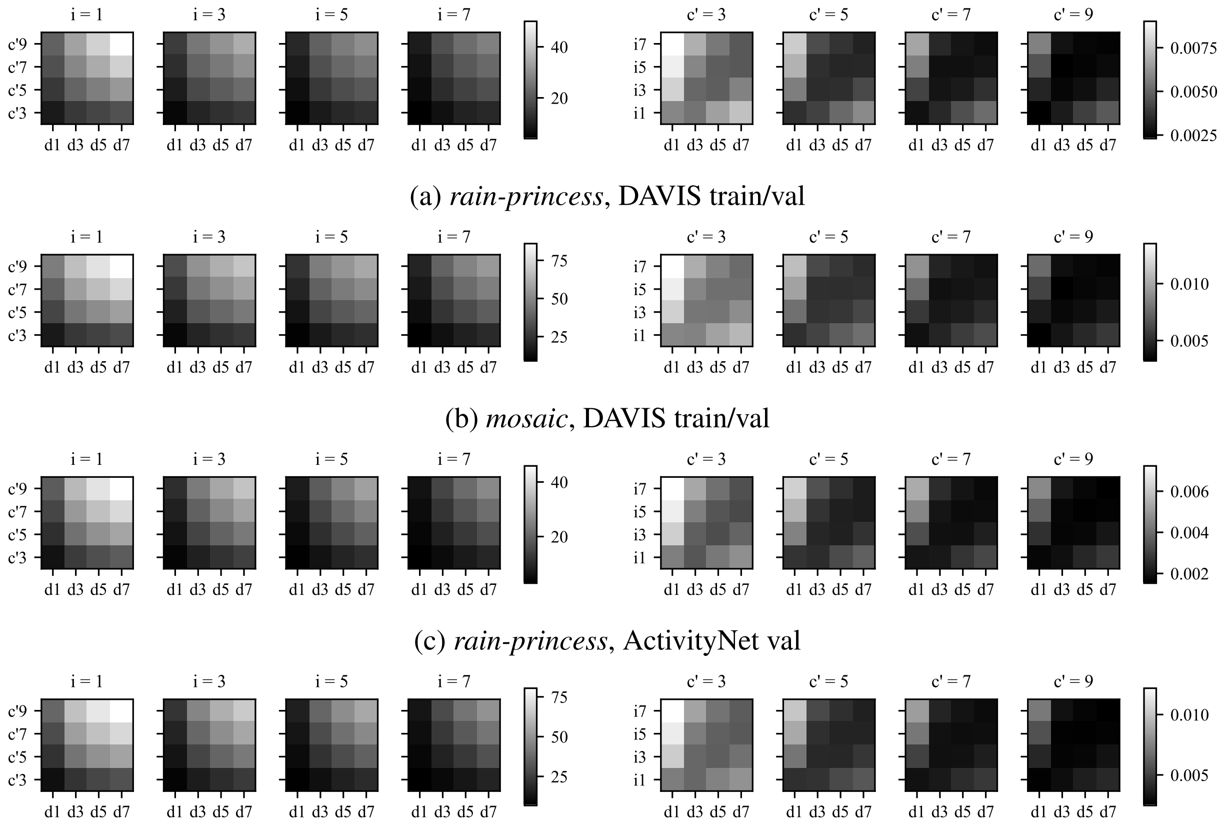
We perform a grid search over our method's hyperparameters to link them concretely to style adherence and temporal consistency. These hyperparameters consist of the number of frames to



(a) Interpolation step (Section 4.3.1)

(b) Temporal aggregation step (Section 4.3.3)

Figure 4.4: Visualization of the hyperparameters included in our grid search (Section 4.4.3).



(a) *rain-princess*, DAVIS train/val

(b) *mosaic*, DAVIS train/val

(c) *rain-princess*, ActivityNet val

(d) *mosaic*, ActivityNet val

Figure 4.5: Ablative analysis plots. Left plots show FID (style adherence) and right plots show E_{warp} (temporal consistency). Lower is better.

insert between each pair i, the total number of interpolated frames to aggregate for each output frame $c' = 2\gamma + 1$, and the spacing between aggregated interpolated frames d (Figure 4.4). We quantify each setting's performance on the DAVIS and ActivityNet validation sets and present their trends in Figure 4.5.

Lai et al. [42] observe that style adherence and temporal consistency are competing objectives; our hyperparameter grid search supports this claim. Specifically, FID decreases with more interpolated frames, fewer aggregated frames, and a smaller dilation rate, indicating that style adherence is

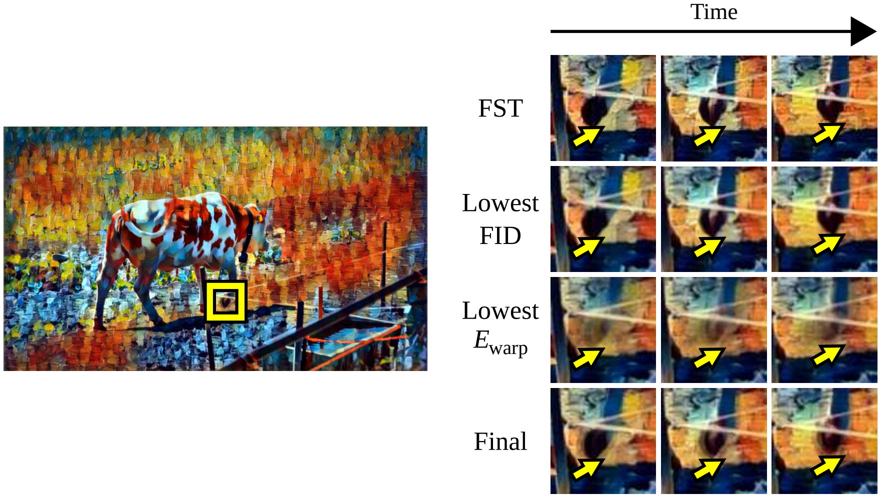


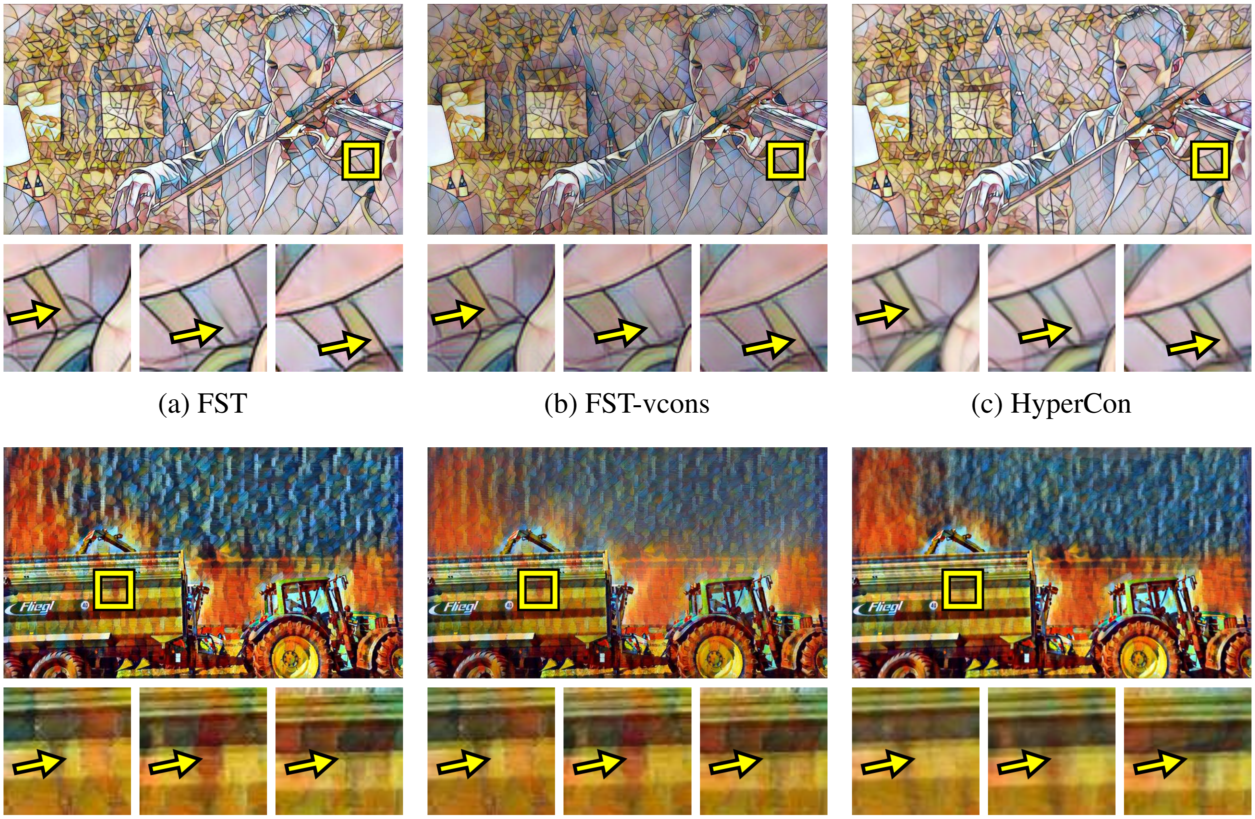
Figure 4.6: Qualitative comparison between frame-wise style transfer (FST) and ablative variants of HyperCon. The “Lowest FID” model reproduces flickering artifacts (e.g., the changing tone near the arrow), while the “Lowest E_{warp}” model overly blurs predictions. Our final model adheres to the intended style without overly blurring predictions.

maximized when HyperCon aggregates over few frames that are very similar to the reference frame. On the other hand, E_{warp} decreases gradually with more aggregated frames, and follows the trend of a parabolic cylinder given fixed c', i.e., there is a sweet spot for i given fixed d and vice-versa. The trends of E_{warp} suggest that temporal consistency is maximized when many frames are aggregated over some effective frame rate. Due to the inherent trade-off between style adherence and temporal consistency, our hyperparameter selection process considers the strongest models in the former criterion and chooses the one that best satisfies the latter; specifically, among the models ranked in the top 15% by FID, we select the one that minimizes E_{warp}.

Turning to qualitative results in Figure 4.6, we confirm that a low FID indicates strong style adherence and that a low E_{warp} indicates strong temporal consistency. However, strictly minimizing one or the other does not yield the most visually satisfying results. For instance, the hyperparameters that minimize FID exhibit the same flickering artifacts as frame-wise style transfer (FST)—observe that the region next to the cow’s foot suddenly changes from blue to orange in the final frame for both FST and the lowest FID model. Meanwhile, the hyperparameters that minimize E_{warp} yield blurry predictions, which is the result of overly smearing several intermediate predictions across frames. Our final model produces consistent tones without overblurring, indicating that our selection strategy sensibly compromises between the two objectives.

4.4.4 Comparison To Prior State-of-the-Art

Now we compare HyperCon to frame-wise style transfer (FST) [86] and a state-of-the-art video consistency method from Lai et al. (FST-vcons) [42]. For FST-vcons, we first apply FST to the



(a) FST (b) FST-vcons (c) HyperCon

(d) FST (e) FST-vcons (f) HyperCon

Figure 4.7: Style transfer comparison for *mosaic* (top) and *rain-princess* (bottom) styles. We show one full frame and crops from three consecutive frames centered at the presented frame. Unlike the baselines, HyperCon draws three consistent lines across the top of the violin (top), and removes the flickering spot on the truck (bottom), thus producing temporally consistent results.

video, and then apply the consistency model of Lai et al. as a post-translation step. Note that FST-vcons must process inputs sequentially, whereas HyperCon can operate on several time steps in parallel due to its short-range dependencies. Given efficient implementations of both methods, HyperCon's parallelizability gives it an advantage in terms of inference time on long videos.

In Figure 4.7, we visually compare FST, FST-vcons, and HyperCon on two DAVIS 2017 test videos. Naturally, FST generates temporally inconsistent predictions by operating on each frame independently: observe the changing arrangement of lines across the top of the violin in Figure 4.7a, as well as the flashing red spot in Figure 4.7d. Meanwhile, FST-vcons has two systematic failures. First, it greatly desaturates predictions, observable across the full frame in Figure 4.7b and in the dulled orange and blue hues of the sky in Figure 4.7e. Second, it leaves inconsistencies intact as a result of darkening regions instead of shifting their hue; for instance, in Figure 4.7b, the pattern of lines on the violin match the inconsistent appearance of FST. HyperCon, on the other hand, properly addresses both of these challenges—in Figure 4.7c, HyperCon maintains a consistent pattern of

		mosaic				rain-princess			
Dataset	Method	FID$^{\downarrow}$	$E_{\text{warp}}{}^{\downarrow}$	$C_{\text{PSNR}}{}^{\uparrow}$	$C_{\text{SSIM}}{}^{\uparrow}$	FID$^{\downarrow}$	$E_{\text{warp}}{}^{\downarrow}$	$C_{\text{PSNR}}{}^{\uparrow}$	$C_{\text{SSIM}}{}^{\uparrow}$
DAVIS 2017	FST [86]	-	0.024829 ± 0.001174	16.72 ± 1.43	0.5166 ± 0.0152	-	0.013934 ± 0.000903	19.75 ± 1.39	0.6030 ± 0.0156
	FST-vcons [42]	24.50	0.010194 ± 0.000500	20.51 ± 1.37	0.5830 ± 0.0140	10.87	0.007071 ± 0.000477	22.73 ± 1.35	0.6717 ± 0.0136
	HyperCon (ours)	**18.04**	**0.008810 ± 0.000493**	**21.04 ± 1.37**	**0.6662 ± 0.0157**	**10.53**	**0.006110 ± 0.000487**	**23.38 ± 1.35**	**0.7480 ± 0.0143**
DAVIS 2019	FST [86]	-	0.029379 ± 0.001684	14.88 ± 0.29	0.4737 ± 0.0184	-	0.017614 ± 0.001309	17.61 ± 0.38	0.5550 ± 0.0189
	FST-vcons [42]	24.89	0.011999 ± 0.000672	18.79 ± 0.29	0.5451 ± 0.0169	14.77	0.008938 ± 0.000644	20.86 ± 0.38	0.6365 ± 0.0159
	HyperCon (ours)	**23.11**	**0.010677 ± 0.000710**	**19.20 ± 0.32**	**0.6105 ± 0.0192**	**13.59**	**0.008117 ± 0.000730**	**21.13 ± 0.43**	**0.6960 ± 0.0175**
ActivityNet	FST [86]	-	0.017895 ± 0.000847	19.29 ± 0.81	0.6478 ± 0.0134	-	0.008428 ± 0.000516	23.45 ± 0.81	0.7469 ± 0.0116
	FST-vcons [42]	26.37	0.006727 ± 0.000311	22.58 ± 0.38	0.7075 ± 0.0115	9.07	0.003923 ± 0.000240	25.97 ± 0.42	0.8008 ± 0.0093
	HyperCon (ours)	**10.09**	**0.005946 ± 0.000298**	**23.61 ± 0.77**	**0.7848 ± 0.0102**	**5.50**	**0.003379 ± 0.000233**	**26.95 ± 0.76**	**0.8544 ± 0.0083**

Table 4.1: Quantitative comparison between frame-wise style transfer (FST) [86], baseline blind video consistency (FST-vcons) [42], and HyperCon (ours) for style transfer. $^{\uparrow}$ and $^{\downarrow}$ indicate where higher and lower is better; bold indicates the best score; and the values after ± are standard error over all videos. FID for FST is blank since this equates to comparing FST to itself. HyperCon obtains lower FID scores than FST-vcons, indicating greater coherence to the intended style of FST, as well as better warping errors and patch-based consistency scores, indicating better temporal consistency.

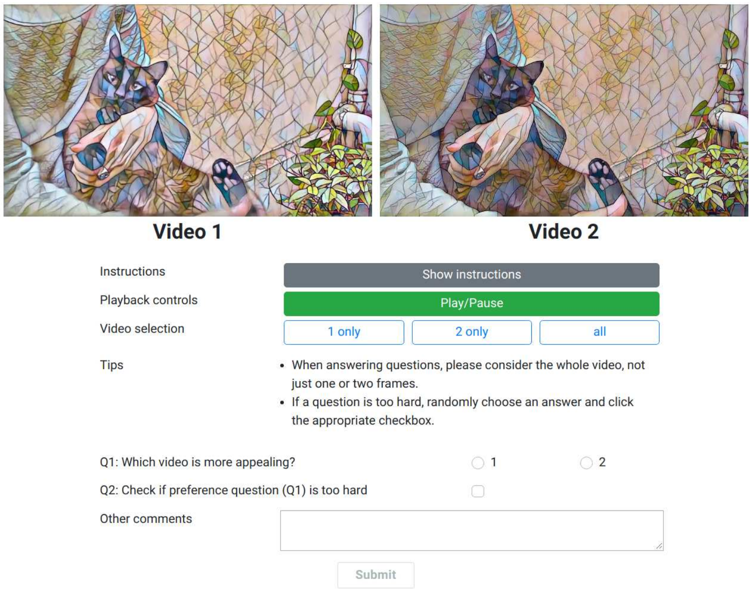
violin lines by passing relevant information between frames, and in Figure 4.7f, it removes the flashing red spot without desaturating the rest of the frame like FST-vcons.

In Table 4.1, we provide a quantitative comparison of our method against the baselines. Since FST stylizes each frame independently, it naturally yields the worst temporal consistency as indicated by its relatively poor warping errors and patch-based consistency scores. Between FST-vcons and our HyperCon approach, the latter obtains better FID scores, warping errors, and patch-based consistency scores, indicating that it is more temporally consistent and better captures the intended style and content compared to FST-vcons.

4.4.4.1 Human Evaluation

To ensure that our quantitative conclusions match those derived from human judgements, we devise a survey on Amazon Mechanical Turk (AMT) to compare HyperCon and the FST-vcons baseline. Specifically, to judge overall quality, we present subjects with two corresponding videos from HyperCon and FST-vcons, and ask which video is more appealing. As for style adherence, we present the two aforementioned videos alongside the frame-wise translated one from FST, and ask method is more similar to FST. For each style, we generate surveys for the 236 videos across our three test sets (Section 4.4.1), and collect five responses for each video-style combination. This yields 5×236 = 1,180 responses per question per style, i.e., a total of 4,720 responses across both questions and both styles.

For our study, we developed custom web interfaces shown in Figure 4.8. The displayed videos are synchronized at all times, and playback can be toggled by pressing the “Play/Pause” button. Users can choose to display one or multiple videos at a time; if one video is selected, it is centered at the top of the page, and if multiple videos are selected, any non-selected video is replaced with a



Instructions

Playback controls

Video selection

Tips

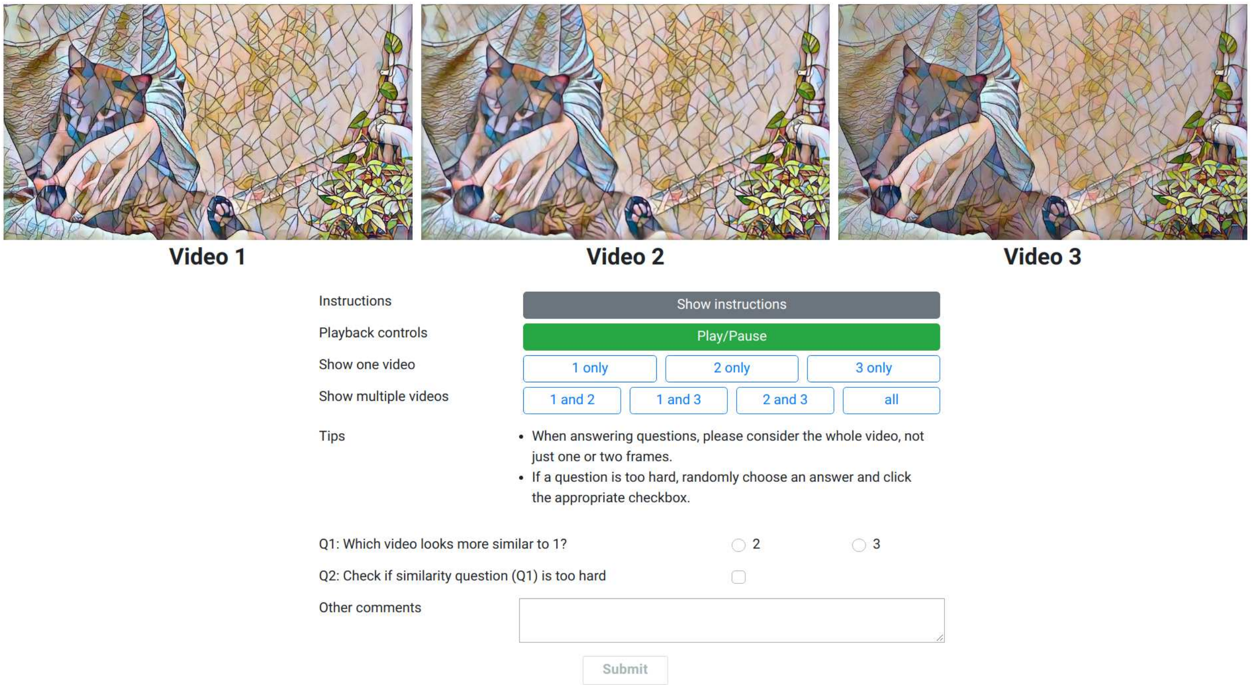
- When answering questions, please consider the whole video, not just one or two frames.
- If a question is too hard, randomly choose an answer and click the appropriate checkbox.

Q1: Which video is more appealing? ○ 1 ○ 2

Q2: Check if preference question (Q1) is too hard □

Other comments

(a) Preferred



Instructions

Playback controls

Show one video

Show multiple videos

Tips

- When answering questions, please consider the whole video, not just one or two frames.
- If a question is too hard, randomly choose an answer and click the appropriate checkbox.

Q1: Which video looks more similar to 1? ○ 2 ○ 3

Q2: Check if similarity question (Q1) is too hard □

Other comments

(b) Style adherence

Figure 4.8: Examples of the interfaces used for the “preferred” and “style adherence” surveys.

black frame. Playback does not reset when different videos are selected, so users can effectively perform A/B-style viewing without losing their place in the video.

Using these interfaces, we asked subjects two questions:

- "Which video is more appealing?" (preferred)

- "Which video looks more to 1?" (style adherence)

For the "preferred" question, we intentionally omitted instructions on what exactly makes a video more appealing since such qualities are inherently ambiguous. As for the "style adherence" question, we framed it in terms of comparing the pairs of videos under evaluation to the frame-wise stylized video as reference; we determined this to be the most concrete, unambiguous way to define the intended video stylization for subjects.

For each combination of style and source video, we randomly selected which method (between HyperCon and the FST-vcons baseline) would appear as the first and second video under evaluation (i.e., videos 1 and 2 in Figure 4.8a and videos 2 and 3 in Figure 4.8b). This forces subjects to watch all videos carefully when answering each question. For the "style adherence" question, the frame-wise stylized video (FST) always appeared as video 1.

We also provided a checkbox to allow the subject to indicate that a question was difficult. We considered an alternative survey formulation that only asked one question, but offered three responses (e.g., "1", "2", or "neither" for the "preferred" survey), but opted for the two-question approach to ensure that rejecting the null hypothesis would lead to an interpretable result.

We paid $0.10 USD per task, which equates to $12 USD/hour if each survey takes 30 seconds to answer. Surveys were generated as separate Human Intelligence Tasks (HITs), so it is not necessarily the case that any given subject saw both questions for any or all videos/styles. In terms of filtering subjects, we did not utilize any demographic-based filters; instead, we created a qualification test utilizing an interface equivalent to Figure 4.8b. The videos used in this test were derived from a source video not included in our DAVIS and ActivityNet test videos, and had a simple variable darkening filter applied to each frame (i.e., multiplying all RGB values by some constant). Examples of the effects we applied include darkening all frames by the same amount, darkening each frame by a random amount, and not darkening any frame. For each video triplet, we always included one of the possible answers as the reference; this effectively forced subjects to match the reference video with one of the other two. We also ensured that the odd video out was blatantly different to make the qualification test straightforward. We generated 10 triplets of qualification test videos, and required subjects to answer correctly for at least 8 of them before working on our tasks.

We provide an aggregate view of our human evaluation results in Table 4.2a. For the "preferred" question, subjects select HyperCon more often than FST-vcons, enough to reject the null hypothesis that subjects select each method with equal probability—in short, they prefer our predictions over

	mosaic		rain-princess	
	Preferred	Style adherence	Preferred	Style adherence
FST-vcons [42]	42.7%	20.9%	44.5%	23.2%
HyperCon (ours)	**57.3%**	**79.1%**	**55.5%**	**76.8%**
p-value	5.53×10^{-7}	0*	1.54×10^{-4}	0*

(a)

	DAVIS 2017 (60 source videos)				DAVIS 2019 (60 source videos)				ActivityNet (116 source videos)			
	mosaic		rain-princess		mosaic		rain-princess		mosaic		rain-princess	
Method	Preferred	Style adherence	Preferred	Style adherence	Preferred	Style adherence	Preferred	Style adherence	Preferred	Style adherence	Preferred	Style adherence
FST-vcons [42]	137	45	128	68	133	43	147	63	234	159	250	143
HyperCon (ours)	**163**	**255**	**172**	**232**	**167**	**257**	**153**	**237**	**346**	**421**	**330**	**437**
Total	300	300	300	300	300	300	300	300	580	580	580	580

(b)

Table 4.2: Human evaluation of style transfer quality. (a) For each style, we list how often subjects favorably select each method across all videos of that style, as well as the p-value of the corresponding χ^2 test. 0* indicates a p-value less than 1×10^{-10}. In all cases, subjects select HyperCon (ours) significantly more often than FST-vcons [42]. (b) A detailed breakdown of responses for each style and dataset.

those of the baseline. Furthermore, for the “style adherence” question, which asks for the video that is more similar to the reference style video, subjects also select our HyperCon method significantly more often; this means that our method is better at preserving the intended style than FST-vcons. This matches the conclusion made from our method’s lower FID scores, and indicates that FID correlates with video quality and style adherence. In Table 4.2b, we provide a detailed breakdown of responses for each question and dataset.

4.5 Experiments: Video Inpainting

To evaluate HyperCon on masked inputs, we apply it to video inpainting. For the frame inpainting subroutine, we re-train the Contextual Attention model from Yu et al. [109] using a modified training scheme that yields higher-quality predictions. Specifically, whereas Yu et al. use the WGAN-GP formulation [133] to update the discriminators of their adversarial loss, we use the original cross-entropy GAN formulation [89] with spectral normalization layers [99] in the discriminator, which stabilize GAN training. Our model achieves a Peak Signal-to-Noise Ratio (PSNR) of 20.41 dB on the Places dataset [54], surpassing the original reported performance of 18.91 dB [109] under the same evaluation setting.

4.5.1 Datasets

To evaluate video inpainting methods quantitatively, we automatically synthesize occlusion masks and use the method under evaluation to inpaint the masked area. As in prior work [55], [61], our masks take the form of a rectangle at a fixed location throughout time. For each video, we randomly select the corner locations of the rectangle, restricting its height and width to lie between 15-50% of the full frame's dimensions (the masks are the same for all evaluated methods). We use the same videos for hyperparameter tuning and evaluation as those described in Section 4.4.1.

4.5.2 Evaluation Metrics

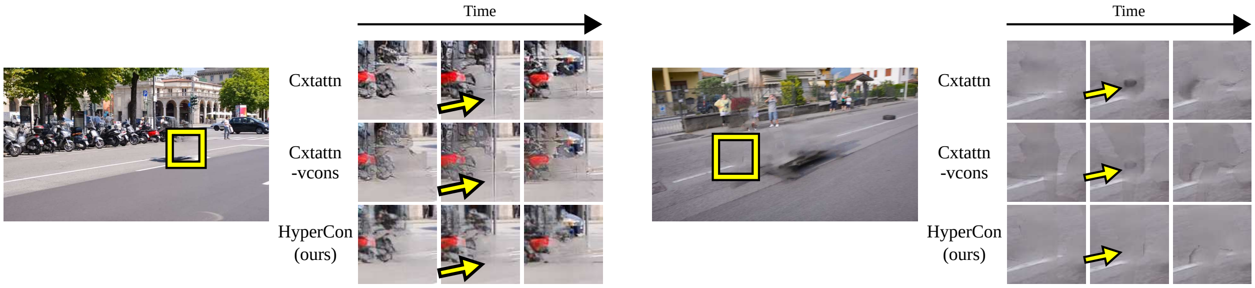
The performance measures that we use for inpainting broadly assess temporal consistency, reconstruction quality, and realism of the composited frames, i.e., the predicted frames with unoccluded pixels replaced by those from the input. We measure temporal consistency using E_{warp} as described in Section 4.4.2. For reconstruction quality, we report mean LPIPS distance (D_{LPIPS}) [43] between corresponding composited and ground-truth (i.e., unoccluded) frames. Although traditional quality measures such as PSNR and SSIM [44] have been used in prior image inpainting work [57], [109], Zhang et al. [43] have shown that LPIPS better correlates with human judgment for paired image comparisons. Additionally, to capture spatio-temporal similarity, we define a new video-based version of LPIPS, VLPIPS, which is a distance between several layers of activations from the I3D action recognition network [64]. We exhaustively compute VLPIPS between corresponding 10-frame segments from the composited and ground-truth videos and report the overall mean. Finally, to evaluate realism, we report FID [41] between the sets of composited and ground-truth frames, as well as a video variant VFID [31] between the sets of 10-frame segments from the composited and ground-truth videos. We compute VFID using the output of I3D's final average pooling layer.

4.5.3 Comparison to Prior State-of-the-Art

For inpainting, we consider two variants of our model: one using a mean filter during temporal aggregation (Section 4.3.3), and the other using a median filter. We compare our models to three state-of-the-art baselines. The first method, Contextual Attention (Cxtattn), inpaints frames independently using the model from Yu et al. [109] with our improved weights (Section 4.5). The second method, Cxtattn-vcons, inpaints frame-wise using Cxtattn, then reduces flickering with the blind video consistency method of Lai et al. [42]. The second phase of Cxtattn-vcons takes the original video and the frame-wise translated video as inputs, which in this case correspond to the masked video (with a placeholder value in the masked region) and the frame-wise inpainted video, respectively. The final method, VINet [56], is a state-of-the-art DNN specifically designed and trained for video inpainting; we use their publicly-available inference code in our experiments. Among these methods, VINet is the only one tasked with inpainting videos at training time, and thus has an advantage over the other methods from having observed natural motion in masked videos.

	DAVIS 2017					DAVIS 2019					ActivityNet				
Method	$D_{\text{LPIPS}}{}^{\downarrow}$	$D_{\text{VLPIPS}}{}^{\downarrow}$	FID$^{\downarrow}$	VFID$^{\downarrow}$	$E_{\text{warp}}{}^{\downarrow}$	$D_{\text{LPIPS}}{}^{\downarrow}$	$D_{\text{VLPIPS}}{}^{\downarrow}$	FID$^{\downarrow}$	VFID$^{\downarrow}$	$E_{\text{warp}}{}^{\downarrow}$	$D_{\text{LPIPS}}{}^{\downarrow}$	$D_{\text{VLPIPS}}{}^{\downarrow}$	FID$^{\downarrow}$	VFID$^{\downarrow}$	$E_{\text{warp}}{}^{\downarrow}$
Cxtattn [109]	0.0457	0.5838	20.94	1.435	0.002186	0.0442	0.5575	**15.55**	1.361	0.002539	$\underline{\mathbf{0.0432}}$	0.5981	$\underline{\mathbf{21.5173}}$	1.4417	0.000894
Cxtattn-vcons [42]	0.0480	0.6076	23.23	1.502	0.001780	0.0478	0.5964	18.52	1.490	0.002166	0.0448	0.6067	23.1642	1.4383	0.000689
VINet [56]	0.0616	0.6062	29.24	1.465	0.001882	0.0539	0.5455	18.22	$\underline{\mathbf{1.195}}$	0.002292	0.0608	0.6139	29.3806	1.3783	**0.000678**
HyperCon-mean (ours)	**0.0450**	**0.5272**	**18.49**	$\underline{\mathbf{1.073}}$	$\underline{\mathbf{0.001540}}$	**0.0437**	**0.5179**	16.07	1.274	$\underline{\mathbf{0.001847}}$	0.0454	$\underline{\mathbf{0.5728}}$	22.5111	$\underline{\mathbf{1.2251}}$	$\underline{\mathbf{0.000640}}$
HyperCon-median (ours)	$\underline{\mathbf{0.0424}}$	$\underline{\mathbf{0.5217}}$	$\underline{\mathbf{17.75}}$	**1.074**	**0.001614**	$\underline{\mathbf{0.0419}}$	$\underline{\mathbf{0.5089}}$	$\underline{\mathbf{15.24}}$	**1.254**	**0.001950**	**0.0441**	**0.5812**	**22.2705**	**1.2601**	0.000683

Table 4.3: Comparison between baseline methods and HyperCon (ours) on the simulated video inpainting task. $^{\downarrow}$ indicates that lower is better. Bold+underline and bold respectively indicate the 1st- and 2nd-place methods. Among the evaluated methods, the HyperCon models consistently place in the top two across all performance measures.

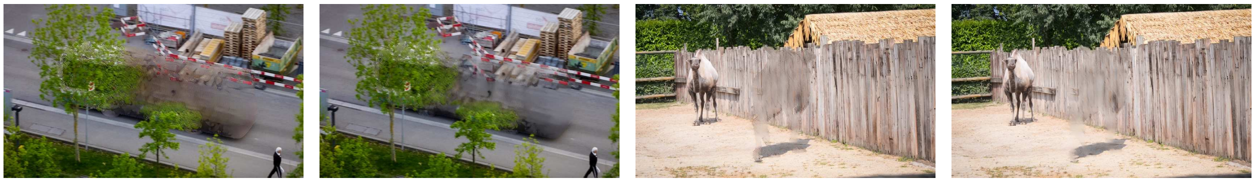


(a) (b)

Figure 4.9: HyperCon substantially reduces flickering compared to the other image-to-video model transfer baselines Cxtattn and Cxtattn-vcons. (a) The pole appears for just one frame with the baselines but not with HyperCon. (b) The gray circle is apparent in the baseline predictions, but not in the HyperCon one.

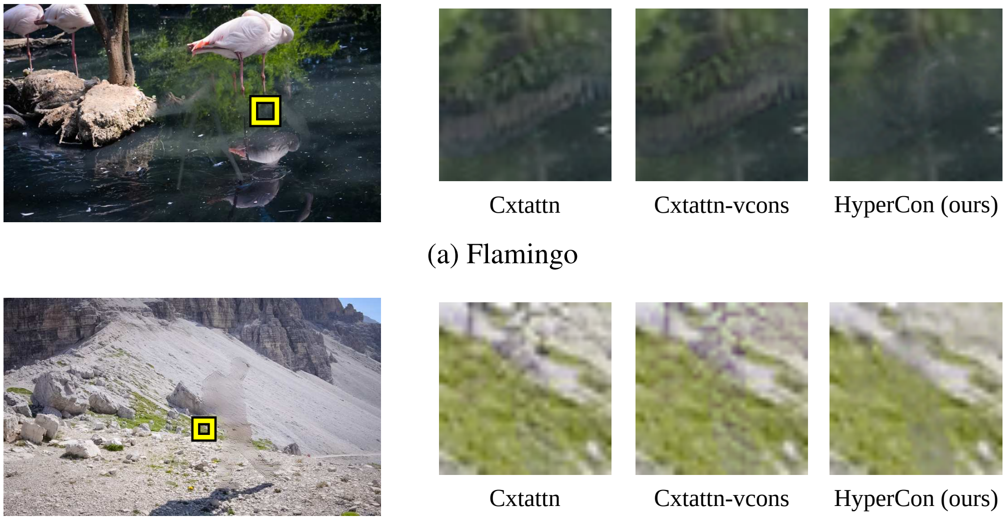
In Table 4.3, we report quantitative results on our test videos for the inpainting task. Across the three test sets, HyperCon-mean obtains the lowest warping error, indicating that its predictions are most consistent with the motion of the ground-truth video (since warping error effectively checks against the estimated flow of the ground-truth video). Between HyperCon-mean and HyperCon-median, the latter generally scores slightly worse in E_{warp} but better in the reconstruction and realism metrics; we suspect that the median filter produces sharper predictions that are more in line with real images, but are harder to match via optical flow (when computing E_{warp}) due to a wider variety of color intensities. Both HyperCon models generally outperform Cxtattn and Cxtattn-vcons across the evaluated datasets and metrics, suggesting that HyperCon is the most reliable method among image-to-video model transfer techniques. Impressively, HyperCon outperforms VINet by a substantial margin in all cases except on DAVIS 2019 under VFID *despite not having seen a single masked video at training time*; this highlights the potential of transferring frame-wise models to video.

Next, we provide a qualitative analysis in the context of real-world object removal for DAVIS 2017 training/validation videos. Comparing our HyperCon method to Cxtattn and Cxtattn-vcons, we found several cases where the baseline methods produce flickering artifacts while ours does



(a) Cxtattn-vcons (b) HyperCon (ours) (c) Cxtattn-vcons (d) HyperCon (ours)

Figure 4.10: Cxtattn-vcons distorts the hue of the inpainted region; HyperCon does not. As a result, our HyperCon predictions blend in more convincingly with the surrounding area.



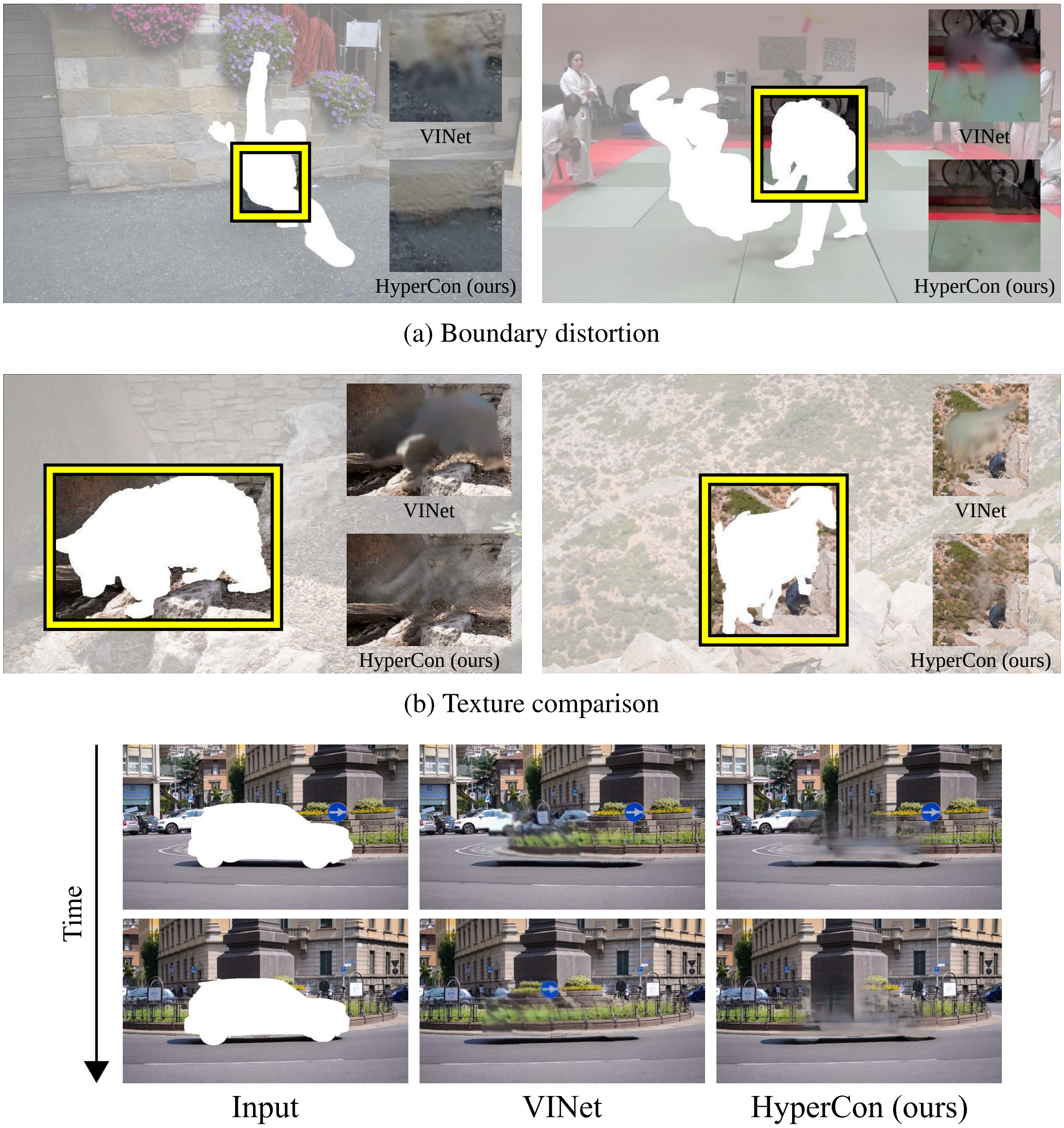
(a) Flamingo

(b) Hike

Figure 4.11: HyperCon generates substantially fewer checkerboard artifacts than Cxtattn and Cxtattn-vcons due to their instability across frames.

not (e.g., the pole in Figure 4.9a and the circle in Figure 4.9b). Additionally, due to the darkening behavior mentioned in Section 4.4.4, Cxtattn-vcons generates darkened predictions that do not blend in with the surrounding region as convincingly as those from HyperCon (Figure 4.10). Furthermore, HyperCon produces the fewest "checkerboard artifacts" [134] since they are temporally unstable and thus get filtered out by HyperCon during aggregation (Figure 4.11).

We conclude our analysis by contrasting HyperCon with VINet [56], the only evaluated method to have seen masked videos during training. We highlight two systematic failures of VINet that HyperCon overcomes: (i) boundary distortion and (ii) textureless prediction. Regarding the first failure, VINet often corrupts its inpainting result over time by incorrectly warping the inpainted structure. For example, in Figure 4.12a, VINet distorts the boundaries of the wall and mat in the shown frames due to the continuous occlusions in those regions; meanwhile, HyperCon successfully produces rigid boundaries. As for the second issue, VINet sometimes initializes the inpainted region with a textureless prediction and fails to populate it with realistic texture throughout the video. In



(a) Boundary distortion

(b) Texture comparison

(c) Negative result for HyperCon

Figure 4.12: Qualitative comparisons between VINet [56] and HyperCon (ours) for video inpainting. (a) VINet distorts the boundaries of the masked wall (left) and practice mat (right), whereas HyperCon successfully recovers them. (b) VINet predicts textureless regions instead of realistic textures; in contrast, HyperCon produces sensible results thanks to its better generalization performance. (c) In this negative result, VINet remembers regions that were unmasked in prior frames, whereas HyperCon incorrectly extends surrounding components from the current frame into the inpainted region.

Figure 4.12b, we compare this behavior with HyperCon, which generates sensible background textures. We posit that HyperCon is better able to hallucinate missing details because it has seen substantially more scenes than VINet during training (i.e., millions of images versus thousands of videos).

The main advantage that VINet has over HyperCon is memory. Thanks to its ConvLSTM unit, VINet is capable of remembering what was behind a masked region for longer periods of time than HyperCon. For instance, in Figure 4.12c, VINet recreates the sign and fence behind the car, while HyperCon propagates the edges of the nearby statue downward. This highlights that memory is a crucial part of properly handling masked regions over long time durations.

4.6 Discussion and Conclusion

In terms of weaknesses, HyperCon does not enforce long-range temporal dependencies beyond the aggregation dilation rate, so it cannot commit to stylization or inpainting choices for the entirety of extremely long videos. In addition, as with other image-to-video model transfer methods, HyperCon is subject to egregious errors of the frame-wise model, which can propagate downstream in certain cases. Integrating the three steps of HyperCon into one trainable, parameter-efficient model could help alleviate these issues by sharing more information between steps and enabling more input frames per unit of memory. Despite these pitfalls, HyperCon is still a promising image-to-video model transfer method for video-to-video translation tasks as seen by its strength in two widely different tasks. We hope that it helps pave the way for further progress in this direction.

CHAPTER 5

Generalization Performance of Video Prediction Models

5.1 Introduction

The question of whether modern video prediction models can correctly represent important sources of variability within a video, such as the motion parameters of cameras or objects, has not been addressed in sufficient detail. Moreover, the manner in which existing video prediction datasets have been constructed—both real-world and synthetic—makes probing this question challenging. Real-world datasets have thus far used videos from action recognition datasets [101], [135] or other "in-the-wild" videos [136], assuring to some degree that the sets of sampled objects and motions will vary realistically between training and testing time. However, as the parameters governing the video appearance and dynamics are unknown, failures in prediction cannot be easily diagnosed. Synthetic datasets, on the other hand, define a set of dynamics (such as object translation or rotation) and apply them to simple objects with pre-defined appearances such as MNIST digits [137] or human head models [138]. Since they are generated from known sets of objects and motion parameters, representations from a video prediction model can be evaluated based on how accurately they predict the generating parameters. However, up until now, both the training and testing splits of these datasets have been generated by sampling from the same object dataset and set of motion parameters, making it difficult to gauge the robustness of video representations to unseen combinations of object and dynamics.

Our proposition is simple: a dataset that explicitly controls for appearance and motion parameters—and can alter them between training and testing time—is essential to answering the question of whether video prediction networks capture in their representations useful physical models of the visual world. To this end, we propose the Moving Symbols dataset, which extends Moving MNIST [139] with features designed to answer this question.[1] Unlike existing implementations of Moving MNIST, which hard-code the image source files and speed of dynamics, we allow these parameters to change between training and testing time, enabling us to evaluate a model's robustness to multiple types of variability in the input. Additionally, we log the motion parameters at each

[1] Moving Symbols was originally proposed by this dissertation's author in [140].

time step to enable the evaluation of motion parameter recovery, making it easier to evaluate model robustness with respect to different combinations of input variability.

To demonstrate the utility of our dataset, we evaluate a state-of-the-art video prediction model from Villegas et al. [93] on a series of dataset splits designed to stress-test its capabilities. Specifically, we present it with videos whose objects exhibit appearances and movements that differ from the training set in a controlled manner. Our results suggest that modern video prediction networks may fail to maintain the appearance and motion of the object for unseen appearances and rates, a problem that, to the best of our knowledge, has not yet been clearly expounded.

5.2 Moving Symbols

The Moving MNIST dataset [139] is a popular toy dataset in the video prediction literature [141]–[143]. Each video in the dataset consists of one or two single digit images that translate within a 64×64 pixel frame with random velocities. Although recent approaches can generate convincing future frames, they have been both trained and evaluated on videos generated by the same object dataset and motion parameters, limiting our understanding of what the networks learn to represent. In particular, it is unclear whether they memorize the object's appearance and update its position over time, or if they learn some low-level shortcut that does not properly capture these attributes.

Our Moving Symbols dataset overcomes this limitation by allowing sampling from *different* object datasets or distributions of motion parameters at training and testing time. For example, the training set may contain slow- and fast-moving MNIST digits while the test set may contain Omniglot characters [144] at medium speed. By adjusting a configuration file, it is trivial to generate paired train/test splits under varying conditions. Furthermore, the video generator logs the image appearance, pose, and rate of movement of each object at each time step, which enables semantically meaningful evaluation as we demonstrate in Section 5.3. This dataset can expose a failure to correctly interpolate video physics, which has been difficult to determine in prior work.

5.3 Experiments

To demonstrate the insights we can gain from the Moving Symbols dataset, we construct two groups of experiments, where each trial consists of different train/test splits (see Table 5.1). Our first set of experiments on *translation speed variation* test whether a video prediction model can generalize the motion seen during training to unseen speeds. We consider three cases: (1a/b) in which the sampled speeds at test time are strictly higher or lower than those sampled at training time, and (1c) in which the training speeds come from a high or low interval while the speeds at test time come from an interval between the two. Our second set of experiments on *appearance*

Experiment group	Train images	Test images	Training rates	Testing rates
(1a) Translation speed variation, slow → fast	MNIST	MNIST	▶	⏭
(1b) Translation speed variation, fast → slow	MNIST	MNIST	⏭	▶
(1c) Translation speed variation, slow/fast → med	MNIST	MNIST	▶ ⏭	⏩
(2a) Appearance variation, MNIST → Icons8	MNIST	Icons8	▶ ⏩ ⏭	▶ ⏩ ⏭
(2b) Appearance variation, Icons8 → MNIST	Icons8	MNIST	▶ ⏩ ⏭	▶ ⏩ ⏭

▶ Slow ⏩ Medium ⏭ Fast

Table 5.1: Experimental setup. The row for each trial describes the objects and motion speeds seen during training and the out-of-domain objects and motion speeds seen during testing.

variation assesses whether video prediction performance worsens when the images observed at test time differ significantly from those seen at training time, while motion dynamics are kept fixed. For these experiments, the training image dataset and the testing image dataset contain vastly different appearances and semantic categories. Specifically, we use MNIST [137] as well as a novel dataset of 5,000 vector-based icons from `icons8.com`. The Icon8 images contain more complex structure than MNIST digits, but omit complex textures such as those in CIFAR-10 [145], which allows us to evaluate the effect of structural complexity on video prediction performance in isolation from textural complexity.

We use our evaluation framework to analyze MCNet [93], a state-of-the-art video prediction model. For each experiment, we generate 10,000 training videos and 1,000 testing videos. We train the model on each unique training set, using ten frames of input to predict ten future frames. After training the model for 80 epochs using the same procedure as Villegas et al. [93], we evaluate it on the out-of-domain test set with ten input frames and twenty predicted future frames.

Figure 5.1 shows a comparison of the predictions made for a random test sequence in each experiment. For the speed variation experiments, we observe that when MCNet is evaluated on out-of-domain videos, it propagates motion by replicating trajectories seen during training rather than adapting to the speed seen at test time (e.g. in experiment (1a) from Table 5.1, MCNet slows down test digits). More surprisingly, the model has difficulty preserving the appearance of the digit when motion parameters change. Since MCNet explicitly takes in differences between subsequent frames as input, it is possible that the portion of the model encoding frame differences inadvertently captures and misrepresents appearance information. In the appearance variation experiments, MCNet clearly overfits to the appearances of objects seen during training. For example, the MNIST-trained model transforms objects into digit-like images, and the Icons8-trained model, which sees a broader variety of object appearances, transforms digits into iconographic images. This may be due to the use of an adversarial loss to train MCNet, which would penalize appearances that are not seen during training. Regardless, the model cannot adapt to observing a new object behaving under familiar dynamics.

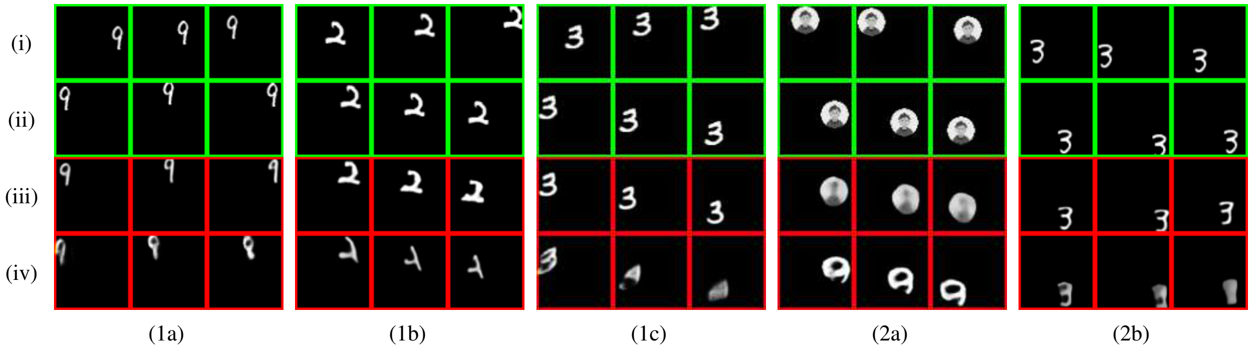


Figure 5.1: Sample predictions for each experiment from Table 5.1. From top to bottom: (i) sample input frames observed by each prediction model at $t = \{3, 6, 9\}$; (ii) sample ground truth frames (unobserved) at $t = \{13, 16, 19\}$; (iii) corresponding predictions from a model tested under the same conditions as the training set; and (iv) predictions from a model tested under different conditions, as per Table 5.1.

To obtain quantitative results, we train an “oracle” convolutional neural network to estimate the digit class and location of MNIST digits in a 64×64 pixel frame and compare them against the logged ground truth. This allows us to draw more interpretable comparisons between generated frames than is possible with common low-level image similarity metrics such as PSNR or SSIM [44]. For the oracle’s architecture, we use LeNet [137] and attach two fully-connected layers to the output of the final convolutional layer to regress the digit’s location. On a held-out set of test digits, the trained oracle obtains 98.47% class prediction accuracy and a Root Mean Squared Error (RMSE) of 3.5 pixels for 64x64 frames.

Figure 5.2 shows our quantitative results on the MNIST-based test sets (all rows except (2a) from Table 5.1). For the speed variation experiments, the frames predicted for out-of-domain test videos induce a large MSE from the oracle, especially during later time steps, whereas in-domain evaluation yields frames that induce a small MSE. This matches our qualitative observation that MCNet fails to propagate the out-of-domain object’s motion, instead adjusting it to match the speeds seen during training. The digit prediction performance of the oracle also drops substantially when evaluating on out-of-domain videos, supporting our observation that MCNet has trouble preserving the appearances of objects that move faster or slower than those seen in training. In the appearance variation experiment (2b), we observe a large digit classification cross-entropy error for the out-of-domain case compared to the in-domain case because the Icons8-trained model transforms digits into iconographic objects. Surprisingly, the model preserves the trajectory of unseen digits better than the MNIST-trained model. This might be because predicting the wrong location for pixel-dense Icons8 images would incur a larger pixel-wise penalty during training than predicting the wrong location for pixel-sparse MNIST images.

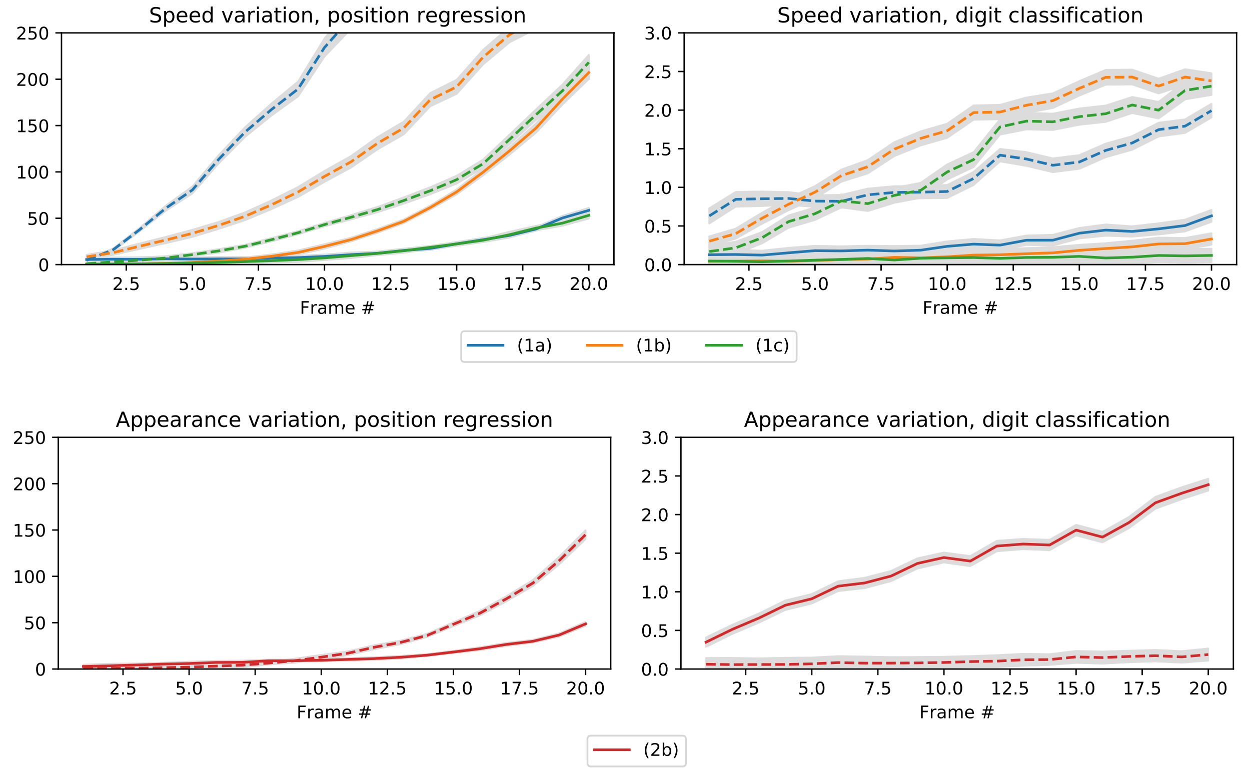


Figure 5.2: Quantitative results to compare in-domain (dotted line) and out-of-domain evaluation performance (solid line), with standard error shown in gray. Across future time steps, we report median values for two metrics: positional Mean Squared Error (MSE) between the oracle's predicted position and ground truth (left plots), and cross-entropy between the oracle's digit prediction and true label (right plots).

5.4 Discussion and Future Work

We have shown that Moving Symbols can help expose the poor generalization behavior of a state-of-the-art video prediction model, which has implications in the real-world case where unseen objects and motions abound. Only a fraction of our dataset's functionality has been demonstrated here—its other features can be used to construct more elaborate experiments, such as introducing scale, rotation, and multiple symbols with increasing complexity, or making use of the pose information as a further supervisory signal. The community's adoption of this dataset as an objective, open standard would lead to a new generation of self-supervised video prediction models. The Moving Symbols dataset is currently available at `https://github.com/rszeto/moving-symbols`.

CHAPTER 6

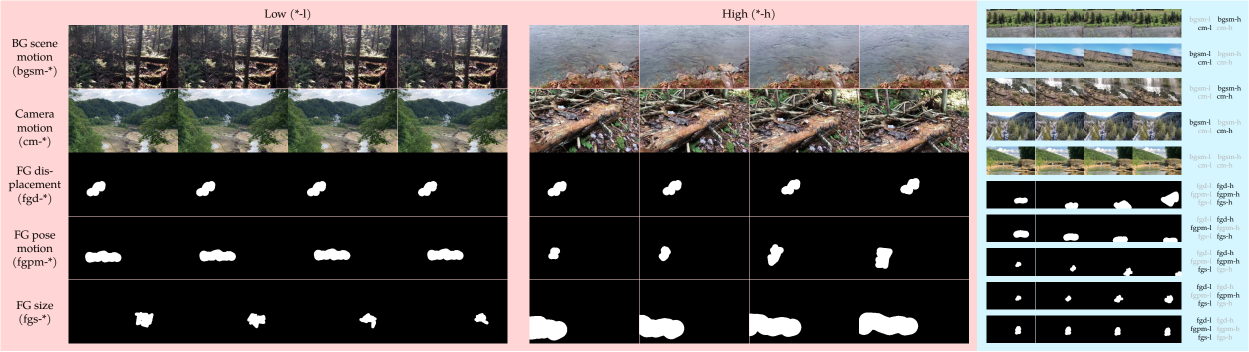
Diagnostic Video Inpainting Benchmark

6.1 Introduction

Video inpainting, i.e., the task of filling in missing pixels in a video with plausible values, pushes the boundaries of modern video editing techniques and enables remarkable applications for film and social media such as watermark and foreground object removal [146], [147]. Compared to image inpainting, video inpainting is more challenging due to the additional temporal dimension, which not only increases the complexity of the solution space, but also places additional constraints on what constitutes a high-quality prediction—in particular, predictions must be coherent in terms of both spatial structure *and* motion. Despite the difficulty of the task, modern results have become quite compelling thanks to the increasing amount of attention that the problem has received as of late [36], [37], [53], [55], [56], [61], [62], [148].

Quantitative evaluation has increased dramatically among recent video inpainting work; however, existing evaluation schemes underemphasize the importance of the contents of the videos and masks used to gauge performance. Typically, video inpainting is evaluated as a reconstruction problem: performance is quantified by masking out arbitrary regions from the video (i.e., "corrupting" it) and scoring the model's ability to recover the masked-out values [37], [52], [61]. However, the difficulty of reconstruction depends on the mask's shape and motion, as well as the content present in the "uncorrupted" video. For example, given a static mask, it is harder to inpaint a video captured by a fixed camera than one captured by a moving camera. In the former case, the region beneath the mask is never visible, so the model effectively needs to "hallucinate" its appearance; in the latter case, the model could transfer appearance information from other frames, a strategy that lies at the heart of many video inpainting approaches [35], [52], [53], [83].

The difficulty of video inpainting is inherently tied to the content of the videos and masks being inpainted; with this principle in mind, we push for more emphasis on content-informed diagnostic evaluation, which can help identify the strengths and weaknesses of modern inpainting methods and improve ablative analysis. To date, the videos used for evaluation have been underappreciated in this regard, having been sourced from datasets for other tasks (e.g., facial analysis [50], [149] and object



(a) DEVIL content attributes

(b) DEVIL dataset w/ annotations

Figure 6.1: A visual overview of our DEVIL dataset. (a) The content attributes that characterize our dataset and are used to create dataset slices for evaluation (i.e., sets of video-mask pairs with a fixed attribute). We label low/high background scene motion or camera motion for videos exhibiting these attribute settings beyond a certain threshold (Section 6.4.2). For occlusion masks, we construct sampling parameters that capture the desired attribute settings and use them to render masks (Section 6.4.3). (b) Videos, masks, and annotations from our dataset. A given video or mask may have multiple attribute labels or none; labels for the same attribute are mutually exclusive (e.g., a mask cannot have both low and high FG displacement).

localization [45], [46]) rather than selected to represent important *inpainting* scenarios. In particular, they contain biases that are essential for the original task, but hinder fine-grained analysis for video inpainting. For example, object localization videos consistently include prominent, moving foreground objects; as a result, standard inpainting evaluation schemes inevitably underrepresent performance on foreground-free videos. Furthermore, other types of motion, e.g., camera and background scene motion, noticeably impact video inpainting performance, but are not controlled for in standard datasets.

In this work, we propose the Diagnostic Evaluation of Video Inpainting on Landscapes (DEVIL) benchmark.[1] It is composed of two parts—the DEVIL dataset and the DEVIL evaluation scheme—which combine to enable a finer-grained analysis than has been possible in prior work. Such granularity is achieved through *content attributes*, i.e., properties of source videos or masks that characterize key failure modes by affecting how easily video inpainting models can borrow appearance information from nearby frames. Specifically, the DEVIL dataset contains source videos labeled with low/high camera and background scene motion attributes, and occlusion masks labeled with low/high foreground displacement, pose motion, and size attributes (Figure 6.1). Meanwhile, the DEVIL evaluation scheme constructs several *slices* of the DEVIL dataset—sets of video-mask pairs in which exactly one content attribute is kept fixed—and summarizes inpainting quality through metrics that capture reconstruction performance, realism, and temporal consistency (Section 6.5.2). By controlling for content attributes and summarizing inpainting quality per attribute

[1] DEVIL was originally proposed by this dissertation's author in [150].

across several metrics, our DEVIL benchmark provides valuable insight into the failure modes of a given inpainting model and how mistakes manifest in the output.

We use our novel benchmark to analyze the strengths and weaknesses of seven state-of-the-art video inpainting methods. By quantifying their inpainting quality on ten DEVIL dataset slices under five evaluation metrics, we provide the most comprehensive and fine-grained evaluation of modern video inpainting methods to our knowledge. Our head-to-head, multi-faceted comparisons allow us to draw several important conclusions. For example, we show that video inpainting methods in which time and optical flow are carefully modeled consistently achieve the best performance across several types of input data. We also show that the relative rankings between methods are highly sensitive to metrics as well as source video and mask content, highlighting the need for comprehensive evaluation. Finally, we show that controlling for source video and mask attributes reveals insightful failure modes which can be traced back to the design of the inpainting method in question.

Our comprehensive diagnostic benchmark enables insightful analysis and serves as an invaluable tool for video inpainting research. To summarize, we provide the following contributions:

- We present the first diagnostic dataset specifically designed for video inpainting to our knowledge, which includes annotations for content-based attributes that represent numerous inpainting failure modes;
- We introduce a novel and comprehensive evaluation scheme that spans ten dataset slices and five evaluation metrics of video inpainting quality;
- We analyze seven state-of-the-art algorithms on our benchmark, providing the most comprehensive quantitative evaluation of video inpainting methods to date; and
- We identify systematic errors among video inpainting methods and highlight directions for future work.

Our code is available at `https://github.com/MichiganCOG/devil`.

6.2 Related Work

6.2.1 Methods

Most video inpainting algorithms borrow visual appearance information from known parts of the video to fill in unknown parts. For example, *alignment-based methods* compute local or global alignments between neighboring frames, and then propagate pixels across aligned locations (alignments are found either through classical feature correspondences [79], [80] or deep neural networks [52], [53], [62]). *Patch-borrowing methods* view videos as spatiotemporal tensors and iteratively paste cuboids or voxels from the known region into the unknown region to maximize global coherence [33], [34], [83]. *Flow-guided methods* propagate visual information along the optical flows estimated between consecutive frames, where the flow for unknown regions is computed iteratively or hierarchically based on known regions to improve performance [35], [56], [61], [148].

Although explicit appearance-borrowing is not leveraged for some methods, e.g., *autoencoder-based methods* [36], [37], [55], [58], we use its prevalence to guide our experimental design. Specifically, we control for content-based attributes that affect the difficulty of borrowing relevant appearance information from other frames, which is possible through our varied data and comprehensive annotations.

6.2.2 Datasets

The source video datasets used in early video inpainting work were small and oriented around qualitative analysis [33], [34], [83]; as a result, it was difficult to compare the performance of early methods across a wide variety of scenarios. Recent video inpainting methods have instead used large-scale datasets—ranging in structure from aligned face videos [50] and driving videos [49] to unconstrained videos from the Internet [45], [46]—to enable more comprehensive analysis. The foreground object segmentation datasets DAVIS [45] and YouTube-VOS [46] have been particularly popular since their annotations can be used to remove the object from the video. Unlike most prior work, we collect novel videos specifically for the inpainting task, emphasizing attributes that affect how well video inpainting models can transfer appearance information across frames. We also use videos of background scenes instead of videos with foreground objects, which enables us to isolate failure modes that are unrelated to foreground motion.

In terms of occlusion masks, static rectangles [55], [61] or foreground masks from other videos [36], [52], [53], [56] are commonly used. Procedural mask generation has also been explored; for instance, Chang et al. [37], [58] render several strokes on a canvas, where each stroke has several control points that randomly move with a certain probability. We extend their work by adding physical constraints for finer control over mask size and motion. Furthermore, we choose our mask sampling parameters based on attributes that directly influence how easily video inpainting models can transfer appearance information across frames.

6.3 Overview of the DEVIL Benchmark

Before introducing the DEVIL benchmark for video inpainting, we first define the task itself. Let $V \in \{0, \dots, 255\}^{H \times W \times 3 \times T}$ be an input RGB video with T frames and a resolution of $W \times H$. V contains a placeholder value for missing voxels (e.g., 0) whose locations are indicated by an input occlusion mask $M \in \{0,1\}^{H \times W \times T}$. Video inpainting aims to produce an inpainted version of V, denoted V^*, with the following characteristics:

- **Reconstruction performance:** V^* is a faithful reconstruction of V^{gt} in the scenario where V is a "corrupted" video derived from some uncorrupted ground truth source video V^{gt}.
- **Realism:** V^* is indistinguishable from a real video.
- **Temporal consistency:** V^* exhibits minimal temporal flickering artifacts.

These criteria are defined more rigorously in Section 6.5.2.

Our DEVIL benchmark is a collection of tools designed to provide a detailed understanding of video inpainting methods and their behavior across a variety of input data. There are two major components of our benchmark: (i) the DEVIL dataset, which contains source videos and occlusion masks that have been specially curated, rendered, and annotated to identify specific failure modes in video inpainting; and (ii) the DEVIL evaluation scheme, which reports a set of quality-based metrics on several "slices" of the DEVIL dataset, each of which represents a particular failure mode.

The DEVIL dataset captures content complexity along five video-level *content attributes*, i.e., properties that affect the difficulty of inpainting a given video-mask pair by influencing the relevance and availability of appearance information from nearby frames. Specifically, it contains source videos with low and high camera and background (BG) scene motion, and occlusion masks with low and high foreground (FG) displacement, pose motion, and size (Figure 6.1a).[2] Furthermore, videos and masks are annotated with these attributes and their settings (low or high) to enable targeted evaluation that controls for their presence. Section 6.4 rigorously defines these attributes and describes our process for collecting videos, masks, and attribute annotations.

Meanwhile, our DEVIL evaluation scheme gauges inpainting quality under multiple *slices* of our dataset, i.e., sets of video-mask pairs characterized by a certain attribute setting. Within each slice, exactly one dataset attribute is kept fixed while the others change freely. By measuring inpainting quality across several slices and metrics, our benchmark provides valuable information on when and how models fail. In Section 6.5, we describe our DEVIL dataset slices and evaluation metrics in further detail.

6.4 The DEVIL Dataset

6.4.1 Collecting Source Videos for the DEVIL

In the context of quantitative evaluation, video inpainting is generally posed as a reconstruction problem [37], [52], [61]; thus, it is useful to evaluate on videos of background scenes without fore-

[2] The term "foreground" (FG) comes from FG object removal applications.

ground objects, where the complete ground-truth background appearance is known (and foreground behavior can be controlled explicitly via occlusion masks). Data from other video understanding tasks do not satisfy this criterion, since they generally feature foreground objects which ground the original task. This is especially true for the two most popular datasets used in video inpainting work, DAVIS [45] and YouTube-VOS [46], which were originally collected for foreground object segmentation.

For this reason, we collect our own videos of background-only scenes, similar to Zhang et al. [36]. In particular, we target scenic landscape videos in which people have filmed natural outdoor locations from both casual and cinematic viewpoints. Because the primary subject of these videos is the background, they are substantially less likely to contain prominent foreground objects, and are thus good targets for curating our source video collection.

To identify a high-quality set of source videos depicting scenic landscapes, we begin by searching Flickr [151] for videos that contain the term "scenic" in their metadata. From these preliminary results, we identify a small number of users who upload a large volume of high-quality, non-post-processed videos. We then refine our search to "scenic" videos from those users within a given upload time frame (January 2017 - January 2019). From these videos, we automatically detect and discard any that contain shot transitions, resolutions not equal to 1920×1080, or COCO object classes [152] as detected by a Mask R-CNN model [153] provided by Detectron2 [154]. After automatic filtering, we manually inspect the remaining videos and remove those that contain undetected foreground objects or shot transitions, as well as other signs of post-processing (e.g., sped-up videos). We split the remaining videos into clips containing between 45-90 frames, which constitute a grand total of 1,250 source clips.

6.4.2 Annotating DEVIL Source Video Attributes

For our DEVIL source videos, we annotate two types of content attributes: camera motion and BG scene motion (Figure 6.1a). **Camera motion** encompasses frame-to-frame differences that are induced by changes in the camera's pose relative to the scene (i.e., camera extrinsics); **BG scene motion** refers to frame-to-frame differences that result from changes in the scene itself, such as running bodies of water or trees that sway due to strong winds (i.e., motion among "stuff" classes in the object detection sense [155]).

We select these attributes for two reasons. First, they represent two sources of complex motion with different low-level characteristics that video inpainting models must replicate well to produce convincing predictions. Second, they impact video inpainting models by influencing the similarity and relevance of appearance information across frames. For instance, high camera motion can reveal or obscure parts of the scene, or otherwise change the scene's appearance due to perspective; high BG scene motion continuously changes the frame-wise appearance of textures.

These attributes are difficult to quantify concretely based on RGB video frames alone; however, it is possible to distinguish extreme examples of low and high motion by visual inspection and proxy estimates. Thus, for a given attribute, we label videos as containing either low or high motion, but only for a small percentage of videos lying at the extreme ends (videos not labeled for the given attribute may still appear in slices that do not control for it). Not only does this reduce label ambiguity, it also magnifies any performance differences caused by changing a given attribute between low and high settings, thereby highlighting failure modes.

To annotate high BG scene motion, we manually identify clips that contain running bodies of water that cover at least 40% of the frame for all frames; for low BG scene motion, we identify clips that contain no running bodies of water (we establish our BG scene motion annotations based on bodies of water since they are prevalent in our data and easy for people to identify by visual inspection). We did not use automatic classifiers for this attribute due to their poor performance and the automatic bias that would have been introduced through their usage.

To annotate camera motion, we use classical affine alignment techniques and measure the amount of invalid pixels introduced via warping as a proxy for camera motion. The intuition behind this classifier is that high camera motion produces frames with poor pairwise affine alignments, and that warping frames by such transforms introduces a high percentage of invalid pixels into the field of view (the converse is true for low camera motion). Despite the simplicity of this approach, we found that it achieves a sufficiently high precision-recall AUC for our purposes (0.90 on a manually-annotated version of the DAVIS train/val set [45]).

Concretely, we label camera motion as follows: between a given pair of video frames, we first compute bidirectional robust affine transformations using RANSAC [156] over matched SURF keypoints [157]. Then, we warp the frames by the corresponding affine transformation and compute the number of invalid pixels introduced by the warp; we define the inverse of this quantity as the pairwise compatibility between the given frames. For a given clip, we sample ten evenly-spaced frames and compute the minimum pairwise compatibility between all pairs, which we define as the total frame compatibility of the clip. Finally, we obtain camera motion annotations by thresholding the total frame compatibility.

6.4.3 DEVIL Masks and Attributes

For occlusion masks, we consider three attributes that influence the availability of relevant appearance information in nearby frames (Figure 6.1a):

- **FG displacement:** How much the mask's centroid moves over time with respect to the field of view;

- **FG pose motion:** How much the shape of the mask changes over time with respect to the field of view (independent of the displacement of its centroid); and

- **FG size:** The average number of pixels that are occupied by the mask per frame.

Masks with high FG displacement or pose motion reveal complementary parts of the scene over time, whereas FG size explicitly determines how much appearance information can be relied on as ground truth.

To generate occlusion masks with our desired DEVIL attributes, we opt for a procedural generation approach inspired by Chang et al. [37], which enables fine-grained control over mask shape and behavior. In their framework, an initial mask shape is generated by sampling control points along a random walk with momentum (i.e., biased toward an initial direction), and then connecting the control points with a stroke of random thickness. The mask is animated by moving all control points with a given velocity and then slightly perturbing their positions at each time step.

We extend the code of Chang et al.with several changes to enable even finer control over mask size and motion. For example, we reduce the impact of momentum in the initial mask-drawing phase to increase the diversity of mask shapes. Additionally, we apply inward-facing acceleration to the control points whenever they are sufficiently far from the mask's centroid, which effectively constrains its maximum possible area. Furthermore, we force the control points to bounce off the edge of the frame to prevent them from leaving the field of view. Finally, we randomly reverse the temporal dimension of masks with a 50% probability since they tend to grow in size over time.

Because the mask generation procedure is parameterized, we can produce occlusion masks that correspond to our desired DEVIL attribute settings by sampling from distinct configurations. To generate masks with small and large FG sizes, we sample from two corresponding ranges of stroke widths, and also change the maximum possible distance of each control point to the centroid. To vary the FG displacement, we sample the initial velocity of the overall mask from two different ranges. Finally, to generate masks with low and high pose motion, we vary the stochasticity of the control points (i.e., for low pose motion masks, the control points are less likely to accelerate in a random direction per frame).

6.5 The DEVIL Evaluation

6.5.1 Slices of the DEVIL Dataset

The naïve way to evaluate on the DEVIL dataset would be to randomly sample a test set of paired source videos and occlusion masks without accounting for their attributes; however, this provides little insight into the failure modes that cause prediction errors for a given method. Instead, we control for one attribute at a time to isolate its impact on the prediction. Specifically, for each attribute setting, we construct *slices* of the DEVIL dataset, i.e., pre-determined sets of video-mask pairs where the given attribute is fixed as low or high and the others are uncontrolled. By reporting performance on each slice separately, our benchmark can highlight failure modes with finer granularity.

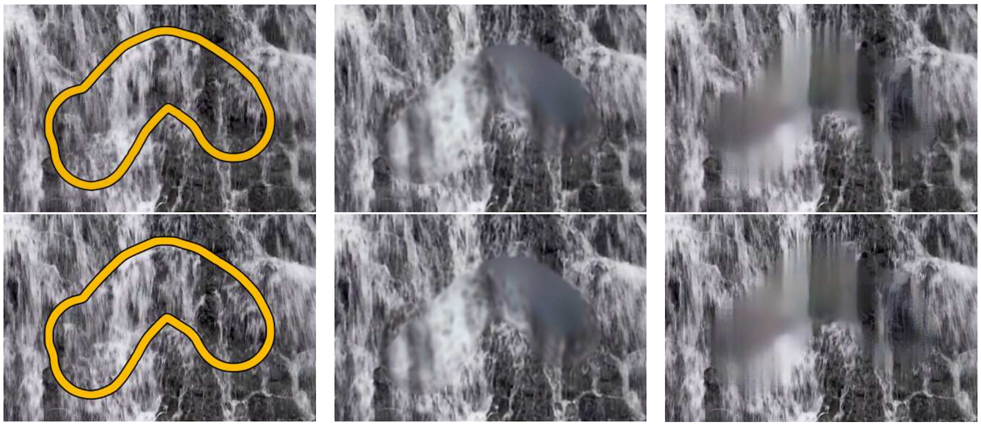
We construct the DEVIL dataset slices as follows. Given the desired attribute setting (e.g., low camera motion), we randomly sample either 150 source videos or masks with that setting (recall that an attribute applies exclusively to either the source video or the mask modality). Then, within the other modality, we sample 150 instances from all available DEVIL instances (e.g., for the low camera motion slice, we sample all rendered DEVIL masks). Finally, we pair together the selected source videos and masks.

6.5.2 Evaluation Metrics

We evaluate on composited inpainting results (i.e., the known region is composited over the inpainting prediction). Inputs are resized to 832×480 resolution; for methods that cannot consume this resolution, we apply mirror padding to both the source video and the mask, run the method, and crop out padded regions from the result for fairness. We quantify performance along three axes of inpainting quality—reconstruction, realism, and temporal consistency—as described in the remainder of this section.

Reconstruction captures the extent to which a video inpainting method predicts the original content in a given reference video (i.e., the version of the video without the occlusion mask). We report two reconstruction metrics: the Learned Perceptual Image Patch Similarity (LPIPS) metric [43] with a pre-trained AlexNet backbone [158], and our own video-based variant called the Perceptual Video Clip Similarity (PVCS) metric with a pre-trained I3D backbone [64]. These metrics measure the distance between corresponding features of a deep neural network. LPIPS is computed between corresponding frames of the reference and inpainted video, whereas PVCS is computed between corresponding 10-frame clips in a sliding window.

Realism indicates how well the inpainting result resembles the appearance and motion observed in a reference set of real videos, independent of the original video from which the input is derived. Unlike reconstruction, realism enforces a smaller penalty for deviating from the original content as long as the prediction exhibits sensible appearance and motion. For our image-based realism metric, we use the Fréchet Inception Distance [41] (FID), which fits multivariate normal distributions over the feature activations of two sets of images and measures their distance; in our case, the two sets correspond to all predicted frames and all reference frames. We also report the video-based equivalent Video FID [56] (VFID), which corresponds to the sets of all inpainted videos and all reference videos (the features are extracted from a pre-trained I3D backbone [64]).



(a) Ground truth (b) VINet (c) CPNet

Figure 6.2: High temporal consistency may indicate overly blurry predictions if reconstruction or realism performance is low (as shown in the results of VINet and CPNet). The area to be inpainted is outlined in yellow in (a).

Temporal consistency measures the proliferation of flickering artifacts, i.e., how much colors at corresponding points of the scene change between consecutive frames. We adapt the patch consistency metric, denoted PCons, from Gupta et al. [21]: for each frame, we extract the 50×50 patch at the centroid of the occlusion mask, compute the maximum Peak Signal-to-Noise Ratio (PSNR) between this patch and neighboring patches in the next frame, and average the result across all frames. Note that stronger temporal consistency is not always ideal: low-quality predictions, such as constant-color or blurry inpainting results, can produce high temporal consistency scores (see Figure 6.2).

6.6 Experiments

To demonstrate the utility of our DEVIL benchmark, we analyze the performance of seven representative state-of-the-art video inpainting methods, using the publicly-available code, model weights, and default runtime arguments provided by the original authors:

- **Joint optimization of flow and color (JointOpt) [35]:** Alternates between optimizing an optical flow estimate and finding suitable patches along the flow. Among our methods, this is the only non-deep learning one.
- **VINet [56]:** Recurrently predicts the next frame by warping intermediate features spatially via optical flow.
- **Deep Flow Completion Network (DFCNet) [61]:** Predicts the optical flow of the video, then inpaints the missing region by propagating known values along the flow.

		Low			FG displacement			High		
LPIPS ▼	PVCS ▼	FID ▼	VFID ▼	PCons ▲	Method	LPIPS ▼	PVCS ▼	FID ▼	VFID ▼	PCons ▲
0.00504	0.1910	7.73	0.0513	36.96	JointOpt	0.00206	0.0995	**1.71**	0.0197	36.07
0.00643	0.3053	21.68	0.1064	47.92	VINet	0.00370	0.2258	8.29	0.0648	44.39
0.00479	**0.1822**	**7.29**	0.0507	**55.81**	DFCNet	**0.00190**	0.1028	2.08	0.0241	**56.35**
0.00419	0.2254	12.72	0.0740	42.36	CPNet	0.00236	0.1462	4.85	0.0367	40.33
0.00379	0.1887	**7.29**	**0.0466**	35.14	OPN	0.00253	0.1474	3.46	0.0305	34.60
0.00411	0.2265	8.18	0.0660	39.90	STTN	0.00293	0.1602	3.68	0.0390	38.39
0.00465	0.1914	9.17	0.0495	37.02	FGVC	0.00200	**0.0986**	1.89	**0.0182**	38.15
0.00471	0.2158	10.58	0.0635	42.16	Mean	0.00250	0.1401	3.71	0.0333	41.18

		Low			FG pose motion			High		
LPIPS ▼	PVCS ▼	FID ▼	VFID ▼	PCons ▲	Method	LPIPS ▼	PVCS ▼	FID ▼	VFID ▼	PCons ▲
0.00401	0.1643	5.79	0.0419	36.84	JointOpt	0.00296	0.1266	3.85	0.0282	36.56
0.00618	0.2869	17.00	0.0956	46.40	VINet	0.00435	0.2456	10.51	0.0730	46.03
0.00354	**0.1604**	**5.06**	0.0430	**54.07**	DFCNet	0.00294	**0.1237**	**3.38**	0.0305	**55.17**
0.00383	0.2060	10.34	0.0655	41.60	CPNet	0.00323	0.1681	6.33	0.0481	41.16
0.00319	0.1794	6.01	0.0433	35.18	OPN	**0.00289**	0.1606	3.68	0.0367	34.54
0.00395	0.2101	6.96	0.0575	39.80	STTN	0.00328	0.1809	4.77	0.0481	39.14
0.00391	0.1634	6.03	**0.0392**	37.58	FGVC	0.00290	0.1261	4.39	**0.0259**	37.75
0.00409	0.1958	8.17	0.0551	41.64	Mean	0.00322	0.1617	5.27	0.0415	41.48

		Low			FG size			High		
LPIPS ▼	PVCS ▼	FID ▼	VFID ▼	PCons ▲	Method	LPIPS ▼	PVCS ▼	FID ▼	VFID ▼	PCons ▲
0.00078	**0.0495**	1.99	**0.0102**	36.18	JointOpt	0.00591	0.2324	**7.30**	**0.0530**	37.19
0.00118	0.1094	6.31	0.0288	43.81	VINet	0.00839	0.4312	23.63	0.1488	50.07
0.00069	0.0519	**0.93**	0.0119	**53.80**	DFCNet	0.00556	**0.2240**	7.71	0.0627	**57.80**
0.00073	0.0680	2.35	0.0155	39.17	CPNet	0.00679	0.3048	13.21	0.0969	43.91
0.00084	0.0563	0.95	0.0119	34.30	OPN	0.00571	0.2874	8.71	0.0708	35.09
0.00113	0.0786	2.02	0.0194	38.24	STTN	0.00645	0.3149	10.89	0.0911	41.12
0.00087	0.0528	1.66	0.0119	37.89	FGVC	**0.00543**	0.2322	9.76	0.0564	38.26
0.00089	0.0666	2.32	0.0157	40.48	Mean	0.00632	0.2896	11.60	0.0828	43.35

		Low			BG scene motion			High		
LPIPS ▼	PVCS ▼	FID ▼	VFID ▼	PCons ▲	Method	LPIPS ▼	PVCS ▼	FID ▼	VFID ▼	PCons ▲
0.00247	**0.1199**	**3.70**	**0.0257**	39.36	JointOpt	0.00512	0.1680	3.89	**0.0333**	34.91
0.00434	0.2549	13.01	0.0791	47.87	VINet	0.00543	0.2793	14.39	0.0796	**47.02**
0.00275	0.1381	5.97	0.0387	**54.00**	DFCNet	**0.00362**	**0.1620**	**3.40**	0.0398	40.00
0.00279	0.1691	6.57	0.0499	43.38	CPNet	0.00398	0.2070	5.24	0.0591	41.48
0.00254	0.1587	4.90	0.0397	36.67	OPN	0.00367	0.1819	4.27	0.0349	33.55
0.00299	0.1873	6.60	0.0569	40.82	STTN	0.00447	0.2030	4.15	0.0438	38.78
0.00314	0.1347	7.70	0.0370	39.16	FGVC	0.00376	0.1697	4.57	0.0356	36.50
0.00300	0.1661	6.92	0.0467	43.04	Mean	0.00429	0.1958	5.70	0.0466	38.89

		Low			Camera motion			High		
LPIPS ▼	PVCS ▼	FID ▼	VFID ▼	PCons ▲	Method	LPIPS ▼	PVCS ▼	FID ▼	VFID ▼	PCons ▲
0.00269	0.1311	**3.03**	0.0253	39.39	JointOpt	**0.00275**	0.1319	**2.36**	0.0220	33.84
0.00610	0.2909	17.87	0.0863	48.39	VINet	0.00467	0.2417	10.44	0.0613	44.00
0.00254	**0.1169**	3.58	**0.0224**	**55.22**	DFCNet	0.00352	0.1353	3.34	0.0293	**45.13**
0.00344	0.1585	5.25	0.0339	44.04	CPNet	0.00496	0.2227	7.21	0.0697	39.56
0.00357	0.1749	3.64	0.0340	34.81	OPN	0.00327	0.1679	4.60	0.0332	32.35
0.00349	0.1521	4.34	0.0280	41.95	STTN	0.00529	0.2410	8.95	0.0783	36.38
0.00284	0.1378	6.28	0.0258	42.65	FGVC	0.00312	**0.1239**	3.05	**0.0206**	34.61
0.00352	0.1660	6.28	0.0365	43.78	Mean	0.00394	0.1806	5.71	0.0449	37.98

Table 6.1: The performance of each inpainting method on each DEVIL slice and evaluation metric. Bold indicates the best method; ▼ and ▲ indicate that lower or higher is better, respectively.

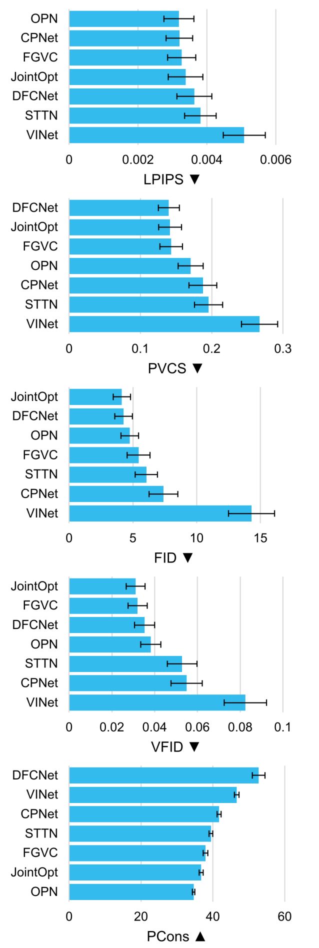


Figure 6.3: Mean performance of each method across all DEVIL slices. Error bars show standard error across the ten slices.

- **Copy-Paste Network (CPNet) [52]:** Estimates affine transformations between frames with a task-driven deep neural network, and then copies features across aligned frames via attention.

- **Onion Peel Network (OPN) [53]:** Iteratively inpaints the exterior of the current missing region by attending to relevant locations in the known region.

- **Spatio-Temporal Transformer Network (STTN) [62]:** Decodes the missing region with a Transformer [159] that consumes multi-scale patches from the entire video.

- **Flow-Edge Guided Video Completion (FGVC) [148]:** Extends DFCNet [61] by leveraging flow between non-adjacent frames and using edge information to solve for piecewise-smooth flow predictions.

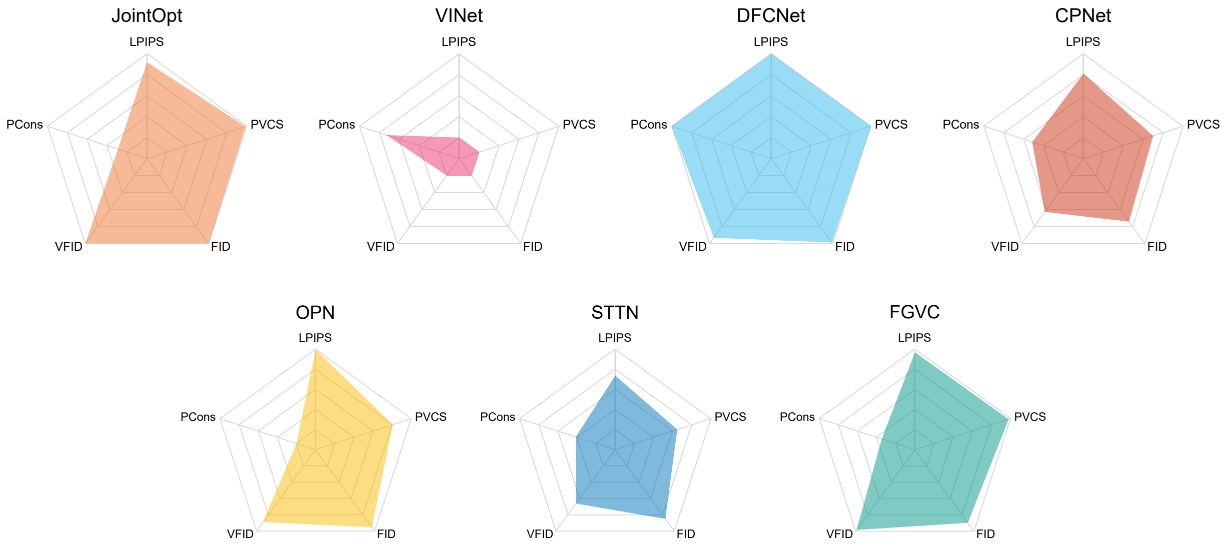


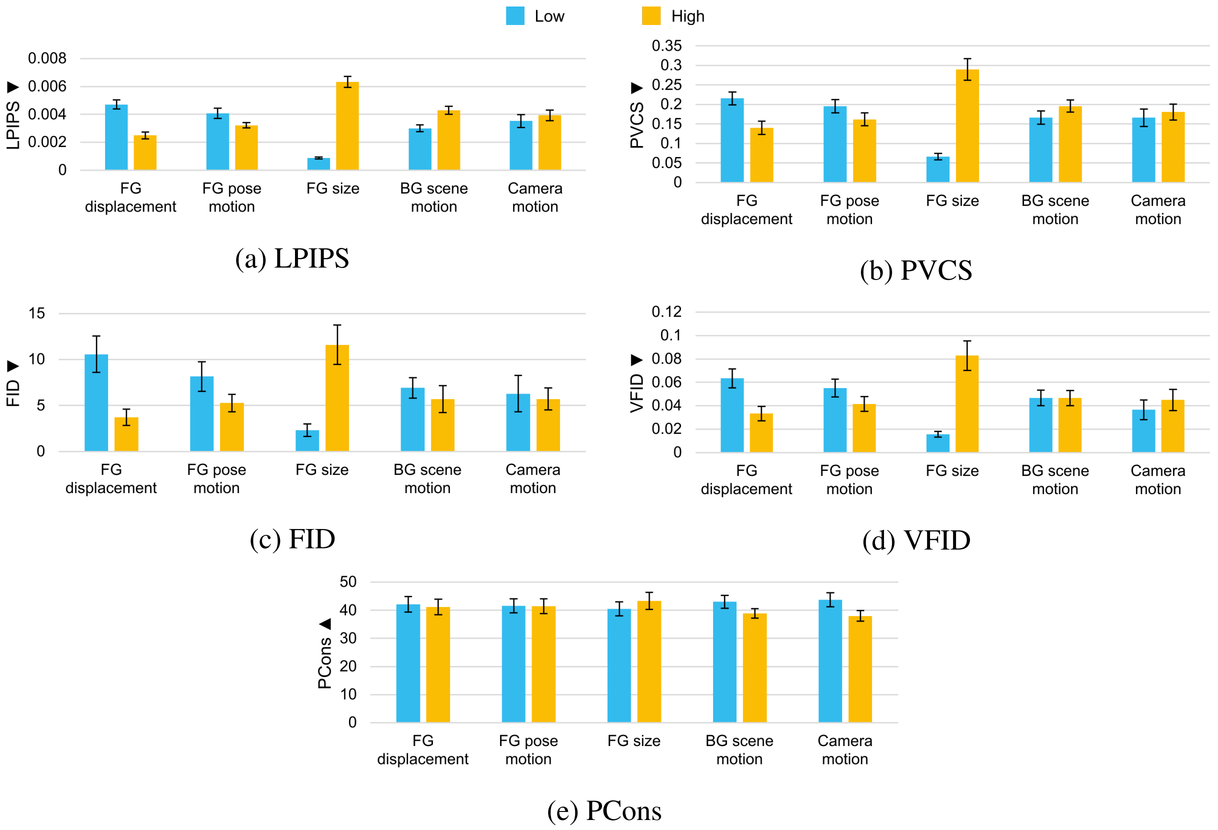
Figure 6.4: Visualizations of each model's performance across the five evaluation metrics averaged over all DEVIL slices; larger area is better. Performance is scaled linearly and independently per metric such that the innermost and outermost pentagons respectively correspond to the weakest and strongest observed mean performance.

6.6.1 Aggregate Analysis

In Table 6.1, we report the performance of all evaluated methods on each DEVIL slice; in Figure 6.3, we compare their performance averaged across all slices. We observe that JointOpt, DFCNet, and FGVC consistently outrank or perform within one standard error of the other methods in terms of the reconstruction metrics LPIPS/PVCS and the realism metrics FID/VFID. They all explicitly solve for the optical flow of the inpainted video during inference, suggesting that *computing task-driven flow is an essential ingredient in producing the highest-quality video inpainting results*. Additionally, JointOpt remains competitive among recent deep learning-based solutions, suggesting that improvements may arise by adapting traditional subroutines, e.g., PatchMatch [84], to deep learning.

The three mid-tier methods—OPN, STTN, and CPNet—borrow intermediate features across distant time steps, but do not use time as an ordered structure. Meanwhile, VINet models time through a recurrent unit, but has the shortest temporal receptive field among the evaluated methods and cannot propagate information from future time steps back through the entire video. These results indicate that *modeling time as a proper ordered structure with long-range dependencies greatly improves inpainting quality.*

In terms of temporal consistency, DFCNet achieves the highest PCons since it propagates pixel values directly along the predicted flow maps of adjacent frames. Interestingly, JointOpt and FGVC



(a) LPIPS

(b) PVCS

(c) FID

(d) VFID

(e) PCons

Figure 6.5: Comparison of DEVIL slice difficulty based on average model performance. ▼ and ▲ indicate that lower and higher is better, respectively; error bars show standard error across the seven evaluated methods. The type of content given at test time, especially the mask content, greatly affects the difficulty of the task.

achieve lower PCons despite also propagating values along the flow, likely due to their ability to transfer candidate values from non-adjacent frames. VINet and CPNet achieve high temporal consistency at the cost of low-quality predictions lacking appropriate texture or motion (Figure 6.2); the behaviors of these two models indicate that temporal consistency is most meaningful when reconstruction and realism performance is also high.

In Figure 6.4, we visualize the strengths and weaknesses of each method. DFCNet is strong and well-rounded, nearly achieving the best performance across all metrics. JointOpt, OPN, and FGVC exhibit strong reconstruction and realism performance, but are relatively weak in terms of temporal consistency. CPNet is well-rounded, but its overall performance is weaker than that of DFCNet. Finally, VINet performs poorly overall, although it achieves decent temporal consistency.

DEVIL Attribute Difficulty We now analyze how DEVIL attributes impact overall inpainting difficulty to highlight their utility in video inpainting evaluation. In Figure 6.5, we compare the difficulty of each DEVIL attribute setting by computing each metric over all methods on the

	FG displacement		Camera motion	
	Low	High	Low	High
LPIPS	**OPN**	**DFCNet**	**DFCNet**	**JointOpt**
	STTN	FGVC	JointOpt	FGVC
	CPNet	JointOpt	FGVC	OPN
	FGVC	CPNet	CPNet	DFCNet
	DFCNet	OPN	STTN	VINet
	JointOpt	STTN	OPN	CPNet
	VINet	VINet	VINet	STTN
	Low	High	Low	High
VFID	**OPN**	**FGVC**	**DFCNet**	**FGVC**
	FGVC	JointOpt	JointOpt	JointOpt
	DFCNet	DFCNet	FGVC	DFCNet
	JointOpt	OPN	STTN	OPN
	STTN	CPNet	CPNet	VINet
	CPNet	STTN	OPN	CPNet
	VINet	VINet	VINet	STTN

Table 6.2: Methods sorted by performance from best to worst based on three variables: the metric (LPIPS/VFID), the attribute (FG displacement/camera motion), and the setting of said attribute (low/high). The strongest method (highlighted in bold) depends on all three variables, showing that no one method dominates our challenging benchmark.

corresponding test slice. As evidenced by the reconstruction and realism metrics, the mask attributes substantially impact the difficulty of video inpainting; in particular, higher FG displacement and pose motion, as well as smaller FG size, lead to better performance. These trends make sense—more relevant appearance information is available in other frames when the occlusion mask is smaller and moves more over time. In contrast, the overall impact of camera and BG scene motion is small due to the differing sensitivities of each method to these attributes (Section 6.6.2). The temporal consistency metric PCons changes less dramatically under the DEVIL attributes (Figure 6.5e), suggesting that temporal consistency performance is relatively stable under changes in the source video and mask content.

6.6.2 Model Sensitivity to DEVIL Attributes

Next, we compare the performance of each method on individual DEVIL slices to demonstrate the impact of video and mask content, as well as the metric, on their rankings. Table 6.2 lists the methods sorted by performance and grouped by three variables: (i) the metric, (ii) the DEVIL attribute, and (iii) the particular setting of the attribute (i.e., low or high). Note that the two attributes shown, FG displacement and camera motion, span the two modalities of the DEVIL dataset (masks and videos, respectively); also note that these three variables span the complexity of the DEVIL evaluation scheme in terms of metrics and dataset slices. Even among the eight different combinations of variables shown out of the possible 50, four methods achieve the best performance for at least one combination; this demonstrates that our DEVIL benchmark is a substantial challenge

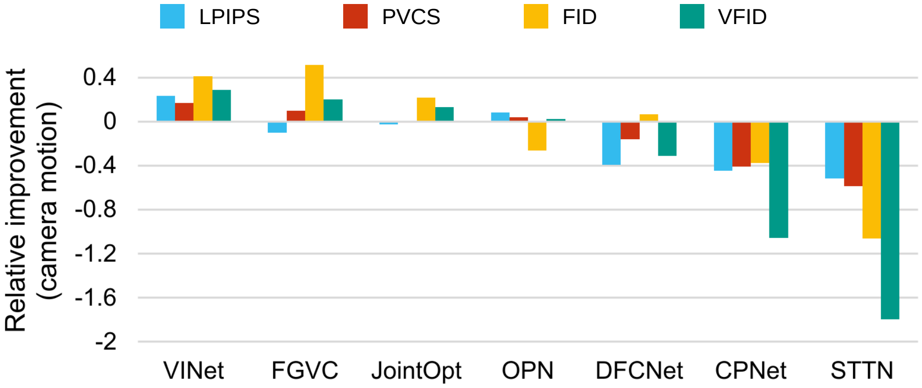
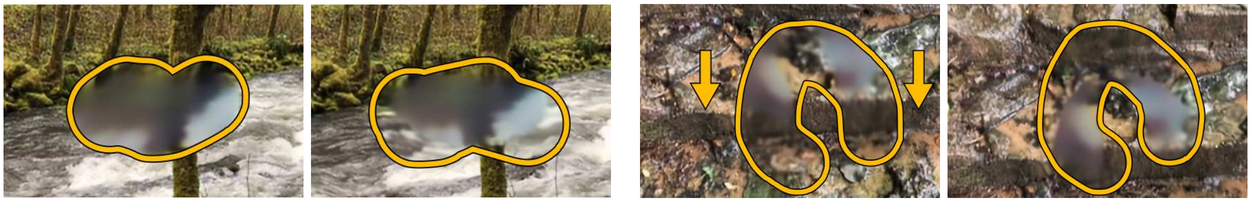


Figure 6.6: Relative improvement of each method under reconstruction and realism metrics when camera motion changes from low to high.



(a) Low camera motion

(b) High camera motion

Figure 6.7: VINet prediction examples. VINet improves with high camera motion since content “moves into” the missing region (arrows show the direction of camera motion).

for video inpainting methods.

Finally, we analyze the sensitivity of the methods to each DEVIL attribute by changing the label and plotting the relative difference in performance. Specifically, for a given attribute, we compute the relative improvement as $(\text{score}_{\text{hi}} - \text{score}_{\text{lo}})/\text{score}_{\text{lo}}$ (or the negation of this value if a lower score is better), where score_{hi} and score_{lo} correspond to model performance on the high and low slices of the attribute, respectively. As evidenced by the following observations, our DEVIL attributes enable a fine-grained comparison of failure modes among different inpainting models.

In Figure 6.6, we see divergent behavior among the methods when camera motion is increased: for example, STTN and CPNet experience dramatic performance drops under all reconstruction and realism metrics, whereas VINet experiences substantial gains. From a design standpoint, the behaviors of STTN and CPNet make sense because they rely on aligning patches or entire frames from other time steps to borrow their features, which can fail catastrophically under heavy camera motion. On the opposite end, VINet’s behavior reflects its tendency to inpaint realistic textures only after they move into the missing portion of the frame (Figure 6.7), which is unlikely to occur without camera motion.

From Figure 6.8, we see that reconstruction performance under LPIPS and PVCS consistently

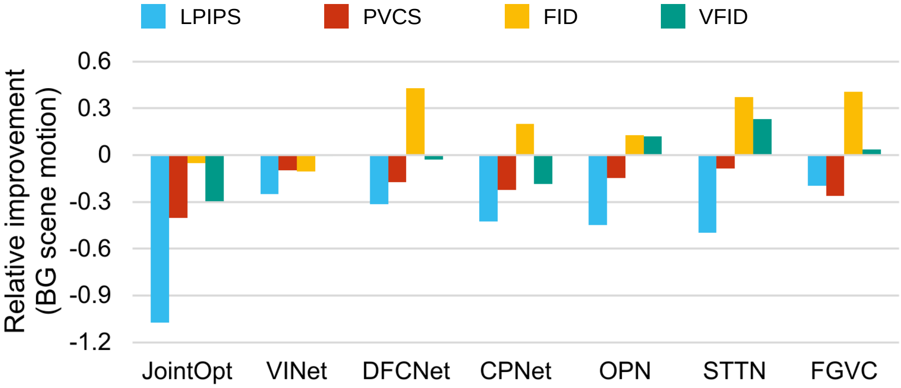
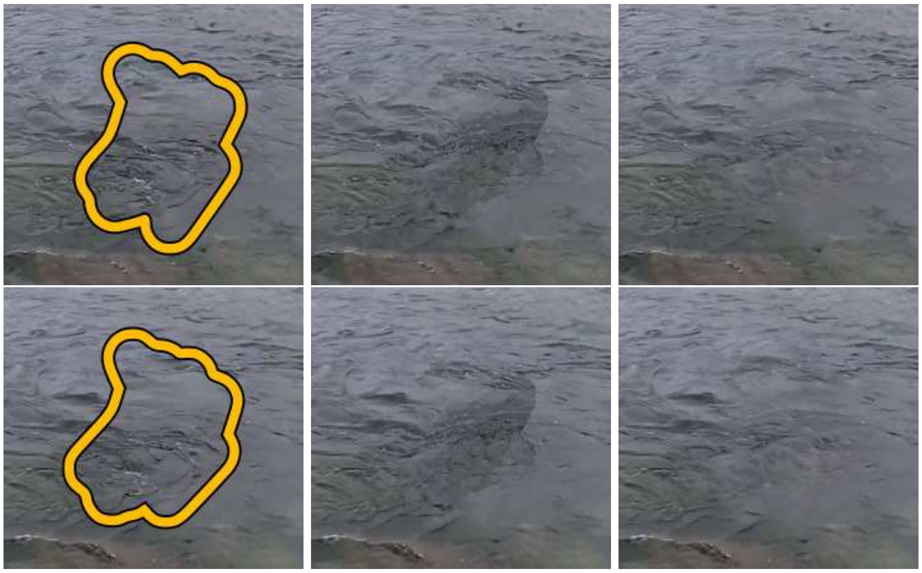


Figure 6.8: Relative improvement of each method under reconstruction and realism metrics when BG scene motion changes from low to high.

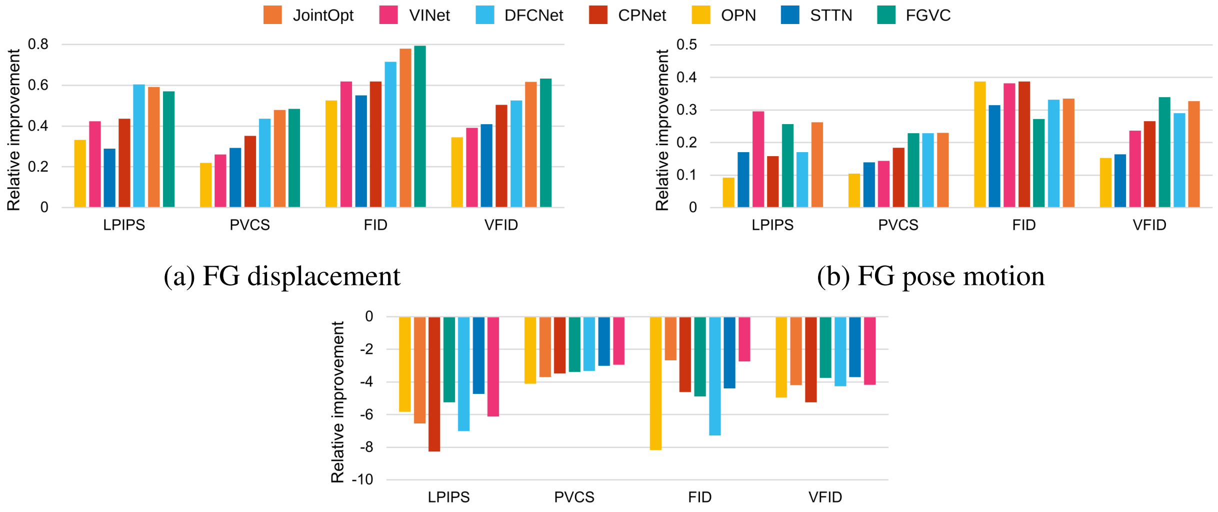


(a) GT (b) DFCNet (c) FGVC

Figure 6.9: Example inpainting predictions from the high BG scene motion slice. For DFCNet and FGVC, the predictions diverge from the original content, but still exhibit semantically sensible appearance.

worsens when BG scene motion increases, reflecting the challenge of replicating complex dynamics precisely. Interestingly, frame realism under FID actually improves dramatically for some methods such as DFCNet and FGVC because they inpaint “alternate realities”, i.e., appearance that diverges from the original content, but still captures the original broad structure (Figure 6.9). Among the methods that explicitly infer and propagate across flow (FGVC, JointOpt, and DFCNet), only JointOpt achieves worse frame realism performance, suggesting that *video inpainting quality can be improved by simply learning flow in a task-driven, end-to-end manner.*

Moving on to mask attributes, Figure 6.10a-b shows that all methods perform better with in-

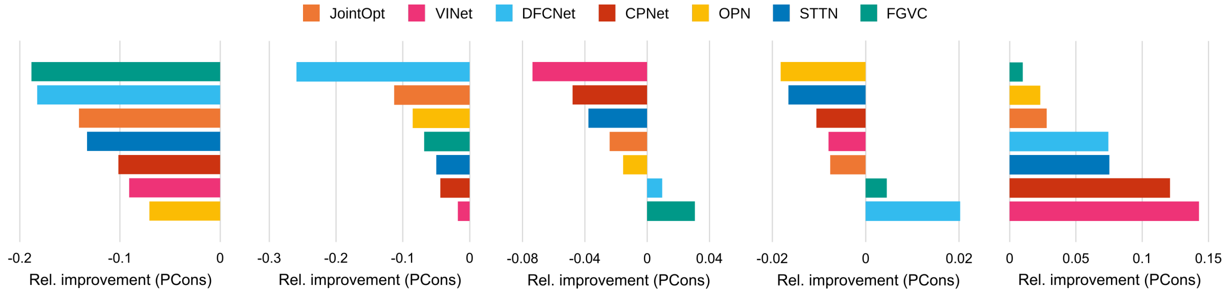


(a) FG displacement

(b) FG pose motion

(c) FG size

Figure 6.10: Relative improvement of each method under reconstruction and realism metrics when DEVIL mask attributes change from low to high. Within each plot, the methods are sorted by PVCS performance.



(a) Camera motion (b) BG scene motion (c) FG displacement (d) FG pose motion (e) FG size

Figure 6.11: Relative improvement in temporal consistency when DEVIL attributes change from low to high.

creased FG displacement and pose motion, indicating that they all leverage the increased availability of background information. The flow propagation methods generally benefit the most, likely due to the explicit transmission of more ground-truth appearance information along predicted flows.

In Figure 6.10c, we observe worse performance when the mask FG size grows, reflecting the inherently greater difficulty of inpainting more values. Although this trend is universal and intuitive, the quantitative difference between methods still lends insight into their failure modes. For example, OPN is most sensitive to the increased mask size because it iteratively inpaints more outer layers of the unknown region based on its own predictions, thereby accumulating error across more iterations.

Figure 6.11 shows the relative change in temporal consistency performance (PCons) when each DEVIL attribute changes from low to high. Overall, we found that temporal consistency is the aspect of inpainting quality that is least sensitive to changes in DEVIL attributes; however, some models still experience more noticeable differences than others (e.g., DFCNet is remarkably sensitive to BG scene motion).

6.7 Discussion

We have presented the DEVIL benchmark for video inpainting and used it to analyze seven state-of-the-art methods, thereby providing the largest fine-grained analysis of video inpainting to our knowledge. By controlling for five content attributes of the source videos and masks used at test time, our analyses have provided novel insight into the behaviors, strengths, and weaknesses of these methods. We plan to develop an evaluation server so that future video inpainting researchers can easily assess their models using DEVIL.

We note that there are a few important limitations of our benchmark—for instance, run time is not a measurement that is considered. In the construction of DEVIL, we chose to emphasize the importance of rigorous, head-to-head visual quality assessment; however, performance versus speed is an important trade-off that impacts a user's decision to apply one approach over others. Measuring run time would enhance the utility of our benchmark to users who want to identify the most appropriate method for their applications. Additionally, our foreground masks do not fully model the behavior of real-world objects, especially since they are superimposed independent of the source scene dynamics. To composite masks into the scene more convincingly, reliable knowledge of the 3D scene and the camera extrinsics would be necessary.

In terms of our analysis, one important consideration is the selection of algorithms that we have evaluated. The seven methods used in this work cover the primary categories of modern video inpainting approaches (i.e., object-based methods, repetitive tensor-based methods, and deep attention-based methods); our observations therefore lend insights that should help steer overall progress in the task. We note that full-frame inpainting methods such as bi-TAI (Chapter 3) are inappropriate since they assume that the input frames are complete. Our analysis also omits the class of two-stage approaches that combine image inpainting with temporal consistency post-processing, e.g., HyperCon (Chapter 4). Given that this class constitutes a minority of video inpainting approaches, we limit our comparative analysis to a brief discussion here, and leave further analysis to future work.

The emphasis of two-stage approaches on highly modular design facilitates improvements that target specific aspects of visual quality, which are assessed in our benchmark. For instance, if a model exhibits low realism but high temporal consistency, it would indicate that more effort should be spent on improving the image inpainting module than the temporal consistency module. As image inpainting and temporal consistency methods improve in parallel, we anticipate a wider adoption of models that combine them, as well as a better understanding of how competitive such models are compared to self-contained video inpainting solutions.

CHAPTER 7

Future Directions and Conclusion

In this dissertation, we have proposed models and diagnostic tools that enforce better visual quality from video inpainting algorithms in terms of realism, temporal consistency, and reconstruction performance. Our models leverage novel temporal contexts to reduce semantic ambiguity and improve visual quality compared to prior work. For instance, bi-TAI (Chapter 3) utilizes an expanded temporal context in the input to inpaint frames with greater coherence; HyperCon (Chapter 4) synthesizes and exploits the temporal context between consecutive input frames to improve temporal consistency. Meanwhile, our diagnostic evaluation tools enable fine-grained, yet scalable analysis of video inpainting model behavior. Specifically, our Moving Symbols dataset (Chapter 5) enables granular control over its appearance and motion statistics, which we use to analyze a state-of-the-art video prediction model and reveal its overreliance on training set priors. Similarly, our DEVIL benchmark (Chapter 6) investigates the fine-grained failure modes of modern video inpainting methods on a larger scale than prior work. It provides a dataset that accounts for influential video and occlusion mask properties, as well as an evaluation scheme that illustrates the effects of these properties on video inpainting along three orthogonal axes of visual quality.

Our work impacts researchers and content creators who need video inpainting models to support their advanced end tasks. Researchers aiming to develop robust algorithms for video inpainting benefit from our publicly available diagnostic tools, which aim to foster the adoption of open, replicable, and fine-grained evaluation in the inpainting community. In addition, consumers of general video manipulation techniques—both content creators and academics alike—benefit from our advanced inpainting models, which enable the production of more impressive and convincing content for their needs.

Despite the advancements presented in this dissertation, several open challenges remain. With regard to inpainting model design, we have seen generalization issues arise from the deployment of trained models that do not adapt to the input (Chapter 5). While large datasets can help deep learning models handle a wide variety of data, the models may also suffer due to inherent training set biases that do not reflect the data provided at test time. Given the complexities of real world scenes and video data, it is unreasonable to expect all test instances to neatly fit within the span of training examples—more concretely, unobserved events are likely to occur in the test video. To manage this challenge, deep learning models need to explicitly adapt to the input data at test time, similarly to how repetitive tensor-based methods adapt the patches to borrow based on the input (Chapter 2). Although Zhang et al. [36] have adopted this idea of test-time optimization by training a deep model solely on the test instance from scratch, improvements could be made by fine-tuning a pre-trained model at test time. Alternatively, repetitive tensor-based methods and deep learning methods could be combined by training deep features that are informed by a downstream, iterative patch-borrowing scheme in an end-to-end manner.

Aside from inpainting model design, characterizing the space of real world videos also presents an exciting future direction. As part of the DEVIL benchmark (Chapter 6), we have identified a limited number of real world video properties that cause inpainting performance to fluctuate (i.e., camera and background scene motion). However, it is likely that several unaccounted factors such as texture complexity, changes in illumination, and overall video quality (e.g., bit rate or reencoding artifacts) also affect performance. Furthermore, it is unreasonable to exhaustively identify every property of video data that impacts inpainting quality, considering the richness of the video medium and the technical challenges of quantifying such properties unambiguously. In this regard, unsupervised machine learning techniques may be helpful since they can identify meaningful variations in data without manual labels. In particular, they could produce low-level video representations whose features correspond to influential factors of video inpainting performance.

Finally, there remains the open problem of evaluating visual quality at scale in a replicable, reliable manner. While the automated video quality metrics used in this dissertation are well-established in prior work (Chapter 2), the extent to which they correlate with human judgement is relatively unknown compared to image-based metrics [43]. On the other hand, human evaluation also has its pitfalls: large-scale evaluation costs a substantial amount of money and human labor, and results between methods may not be comparable if the survey designs or the demographics of those being surveyed are not sufficiently similar. By studying and improving the correlation between automated and human judgements of video quality, researchers would improve the reliability of automated evaluation and, as a result, reduce the cost of trustworthy visual quality assessment.

BIBLIOGRAPHY

[1] *Video Data Shows Changing Youtube Habits*, en. [Online]. Available: https://www.thinkwithgoogle.com/feature/youtube-video-data-watching-habits/ (visited on 05/26/2021).

[2] P. Bump, *How Video Consumption is Changing in 2021 [New Research]*, en-us. [Online]. Available: https://blog.hubspot.com/marketing/how-video-consumption-is-changing (visited on 05/26/2021).

[3] *Celebrating 25 Years of Premiere Pro*, en, Mar. 2017. [Online]. Available: https://blog.adobe.com/en/publish/2017/03/14/celebrating-25-years-of-premiere-pro.html (visited on 05/26/2021).

[4] S. Bernazzani, *The 20 Best Video Editing Apps for 2021*, en-us. [Online]. Available: https://blog.hubspot.com/marketing/best-video-editing-apps (visited on 05/26/2021).

[5] A. Hern, *'I Don't Want to Upset People': Tom Cruise Deepfake Creator Speaks Out*, en, Section: Technology, Mar. 2021. [Online]. Available: http://www.theguardian.com/technology/2021/mar/05/how-started-tom-cruise-deepfake-tiktok-videos (visited on 05/26/2021).

[6] S. Compton, *More and More Women Are Facing the Scary Reality of Deepfakes*, en-US, Mar. 2021. [Online]. Available: https://www.vogue.com/article/scary-reality-of-deepfakes-online-abuse (visited on 05/26/2021).

[7] A. Romano, *Why Reddit's Face-Swapping Celebrity Porn Craze Is a Harbinger of Dystopia*, en, Jan. 2018. [Online]. Available: https://www.vox.com/2018/1/31/16932264/reddit-celebrity-porn-face-swapping-dystopia (visited on 05/26/2021).

[8] M. Kelly, *Distorted Nancy Pelosi Videos Show Platforms Aren't Ready to Fight Dirty Campaign Tricks*, en, May 2019. [Online]. Available: https://www.theverge.com/2019/5/24/18637771/nancy-pelosi-congress-deepfake-video-facebook-twitter-youtube (visited on 05/26/2021).

[9] S. Chandler, "Instagram's Filter Ban Isn't Enough To Stop Rise In Cosmetic Surgery," en, *Forbes*, Oct. 2019. [Online]. Available: https://www.forbes.com/sites/simonchandler/2019/10/23/instagrams-filter-ban-isnt-enough-to-stop-rise-in-cosmetic-surgery/ (visited on 05/21/2020).

[10] J. Chen, "Important Instagram Stats You Need to Know for 2020," en-US, *Sprout Social*, May 2020. [Online]. Available: https://sproutsocial.com/insights/instagram-stats/ (visited on 05/21/2020).

[11] N. Lomas, "Prisma Labs raises $6M for its AI-powered approach to visual editing," en-US, *TechCrunch*, Apr. 2019. [Online]. Available: https://social.techcrunch.com/2019/04/30/prisma-labs-raises-6-7m-for-its-ai-powered-approach-to-visual-editing/ (visited on 05/21/2020).

[12] B. Paris and J. Donovan, *Deepfakes and Cheap Fakes*, en-US, Publisher: Data & Society Research Institute, Sep. 2019. [Online]. Available: https://datasociety.net/library/deepfakes-and-cheap-fakes/ (visited on 05/26/2021).

[13] C. Purdom, *Deep Learning Technology Is Now Being Used to Put Nic Cage in Every Movie*, en-us, Jan. 2018. [Online]. Available: https://www.avclub.com/deep-learning-technology-is-now-being-used-to-put-nic-c-1822514573 (visited on 05/26/2021).

[14] E. Nolan, *Tom Cruise Cleans up in Latest Deepfake Tiktok Video*, en, Section: Culture, May 2021. [Online]. Available: https://www.newsweek.com/tom-cruise-deepfake-tiktok-new-1594525 (visited on 05/26/2021).

[15] D. Mack, *This PSA About Fake News From Barack Obama Is Not What It Appears*, en, Apr. 2018. [Online]. Available: https://www.buzzfeednews.com/article/davidmack/obama-fake-news-jordan-peele-psa-video-buzzfeed (visited on 05/26/2021).

[16] *Media Forensics*. [Online]. Available: https://www.darpa.mil/program/media-forensics (visited on 05/26/2021).

[17] *Deepfake Detection Challenge Results: An Open Initiative to Advance AI*, en. [Online]. Available: https://ai.facebook.com/blog/deepfake-detection-challenge-results-an-open-initiative-to-advance-ai/ (visited on 05/26/2021).

[18] S. Agarwal, H. Farid, T. El-Gaaly, and S.-N. Lim, "Detecting Deep-Fake Videos from Appearance and Behavior," in *IEEE International Workshop on Information Forensics and Security*, Dec. 2020.

[19] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz, “Super SloMo: High Quality Estimation of Multiple Intermediate Frames for Video Interpolation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2018.

[20] M. Ruder, A. Dosovitskiy, and T. Brox, “Artistic Style Transfer for Videos,” in *German Conference on Pattern Recognition*, 2016.

[21] A. Gupta, J. Johnson, A. Alahi, and L. Fei-Fei, “Characterizing and Improving Stability in Neural Style Transfer,” in *IEEE International Conference on Computer Vision*, Oct. 2017.

[22] C. Vondrick, H. Pirsiavash, and A. Torralba, “Generating Videos With Scene Dynamics,” in *Neural Information Processing Systems*, 2016, pp. 613–621.

[23] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, “MoCoGAN: Decomposing Motion and Content for Video Generation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1526–1535.

[24] M. Saito, E. Matsumoto, and S. Saito, “Temporal Generative Adversarial Nets with Singular Value Clipping,” in *IEEE International Conference on Computer Vision*, 2017, pp. 2830–2839.

[25] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos, “Video Super-Resolution with Convolutional Neural Networks,” *IEEE Transactions on Computational Imaging*, vol. 2, no. 2, pp. 109–122, 2016.

[26] J. Caballero, C. Ledig, A. Aitken, A. Acosta, J. Totz, Z. Wang, and W. Shi, “Real-Time Video Super-Resolution with Spatio-Temporal Networks and Motion Compensation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4778–4787.

[27] M. S. M. Sajjadi, R. Vemulapalli, and M. Brown, “Frame-Recurrent Video Super-Resolution,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2018.

[28] M. Haris, G. Shakhnarovich, and N. Ukita, “Recurrent Back-Projection Network for Video Super-Resolution,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2019.

[29] S. Niklaus and F. Liu, “Context-Aware Synthesis for Video Frame Interpolation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2018.

[30] S. Meyer, A. Djelouah, B. McWilliams, A. Sorkine-Hornung, M. Gross, and C. Schroers, “PhaseNet for Video Frame Interpolation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2018.

[31] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro, “Video-to-Video Synthesis,” in *Neural Information Processing Systems*, Dec. 2018.

[32] T.-C. Wang, M.-Y. Liu, A. Tao, G. Liu, B. Catanzaro, and J. Kautz, "Few-Shot Video-to-Video Synthesis," in *Neural Information Processing Systems*, 2019, pp. 5014–5025.

[33] M. Granados, J. Tompkin, K. Kim, O. Grau, J. Kautz, and C. Theobalt, "How Not to Be Seen — Object Removal from Videos of Crowded Scenes," *Computer Graphics Forum*, vol. 31, no. 2pt1, pp. 219–228, 2012.

[34] A. Newson, A. Almansa, M. Fradet, Y. Gousseau, and P. Pérez, "Video Inpainting of Complex Scenes," *SIAM Journal on Imaging Sciences*, vol. 7, no. 4, pp. 1993–2019, 2014.

[35] J.-B. Huang, S. B. Kang, N. Ahuja, and J. Kopf, "Temporally Coherent Completion of Dynamic Video," *ACM Transactions on Graphics*, vol. 35, no. 6, p. 196, 2016.

[36] H. Zhang, L. Mai, N. Xu, Z. Wang, J. Collomosse, and H. Jin, "An Internal Learning Approach to Video Inpainting," in *IEEE International Conference on Computer Vision*, Oct. 2019.

[37] Y.-L. Chang, Z. Y. Liu, K.-Y. Lee, and W. Hsu, "Free-Form Video Inpainting With 3D Gated Convolution and Temporal PatchGAN," in *IEEE International Conference on Computer Vision*, Oct. 2019.

[38] M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine, "Stochastic Variational Video Prediction," in *International Conference on Learning Representations*, Feb. 2018.

[39] E. Denton and R. Fergus, "Stochastic Video Generation with a Learned Prior," in *International Conference on Machine Learning*, Jul. 2018.

[40] D. Jayaraman, F. Ebert, A. Efros, and S. Levine, "Time-Agnostic Prediction: Predicting Predictable Video Frames," in *International Conference on Learning Representations*, Sep. 2018.

[41] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," in *Neural Information Processing Systems*, Dec. 2017.

[42] W.-S. Lai, J.-B. Huang, O. Wang, E. Shechtman, E. Yumer, and M.-H. Yang, "Learning Blind Video Temporal Consistency," in *European Conference on Computer Vision*, 2018.

[43] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 586–595.

[44] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[45] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 724–732.

[46] N. Xu, L. Yang, Y. Fan, J. Yang, D. Yue, Y. Liang, B. Price, S. Cohen, and T. Huang, "YouTube-VOS: Sequence-to-Sequence Video Object Segmentation," in *European Conference on Computer Vision*, 2018.

[47] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial Discriminative Domain Adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[48] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang, "Domain Adaptation Under Target and Conditional Shift," in *International Conference on Machine Learning*, Jun. 2013.

[49] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian Detection: An Evaluation of the State of the Art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 743–761, 2012.

[50] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "FaceForensics: A Large-Scale Video Dataset for Forgery Detection in Human Faces," *arXiv preprint arXiv:1803.09179*, 2018.

[51] N. Xu, L. Yang, Y. Fan, D. Yue, Y. Liang, J. Yang, and T. Huang, "YouTube-VOS: A Large-Scale Video Object Segmentation Benchmark," *arXiv preprint arXiv:1809.03327*, 2018.

[52] S. Lee, S. W. Oh, D. Won, and S. J. Kim, "Copy-and-Paste Networks for Deep Video Inpainting," in *IEEE International Conference on Computer Vision*, 2019, pp. 4413–4421.

[53] S. W. Oh, S. Lee, J.-Y. Lee, and S. J. Kim, "Onion-Peel Networks for Deep Video Completion," in *IEEE International Conference on Computer Vision*, 2019, pp. 4403–4412.

[54] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 Million Image Database for Scene Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Jun. 2018.

[55] C. Wang, H. Huang, X. Han, and J. Wang, "Video Inpainting by Jointly Learning Temporal Structure and Spatial Details," in *AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5232–5239.

[56] D. Kim, S. Woo, J.-Y. Lee, and I. S. Kweon, "Deep Video Inpainting," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5792–5801.

[57] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image Inpainting for Irregular Holes Using Partial Convolutions," in *European Conference on Computer Vision*, 2018.

[58] Y.-L. Chang, Z. Y. Liu, K.-Y. Lee, and W. Hsu, "Learnable Gated Temporal Shift Module for Deep Video Inpainting," in *British Machine Vision Conference*, Sep. 2019.

[59] T. Judd, F. Durand, and A. Torralba, "A Benchmark of Computational Models of Saliency to Predict Human Fixations," Massachusetts Institute of Technology, Technical report, Jan. 2012. [Online]. Available: `https://dspace.mit.edu/handle/1721.1/68590`.

[60] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[61] R. Xu, X. Li, B. Zhou, and C. C. Loy, "Deep Flow-Guided Video Inpainting," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3723–3732.

[62] Y. Zeng, J. Fu, and H. Chao, "Learning Joint Spatial-Temporal Transformations for Video Inpainting," in *European Conference on Computer Vision*, 2020.

[63] D. C. Dowson and B. V. Landau, "The Fréchet Distance Between Multivariate Normal Distributions," *Journal of Multivariate Analysis*, vol. 12, no. 3, pp. 450–455, Sep. 1982.

[64] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," in *IEEE Conference on Computer Vision and Pattern Recognition*, Jul. 2017.

[65] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization," *SIAM Review*, vol. 52, no. 3, pp. 471–501, Aug. 2010.

[66] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor Completion for Estimating Missing Values in Visual Data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 208–220, Jan. 2013.

[67] C. Mu, B. Huang, J. Wright, and D. Goldfarb, "Square Deal: Lower Bounds and Improved Relaxations for Tensor Recovery," in *International Conference on Machine Learning*, Jun. 2014.

[68] F. L. Hitchcock, "The Expression of a Tensor or a Polyadic as a Sum of Products," *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927.

[69] L. R. Tucker, "Some Mathematical Notes on Three-Mode Factor Analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[70] Z. Zhang, G. Ely, S. Aeron, N. Hao, and M. Kilmer, "Novel Methods for Multilinear Data Completion and De-noising Based on Tensor-SVD," in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014.

[71] M. E. Kilmer, K. Braman, N. Hao, and R. C. Hoover, "Third-Order Tensors as Operators on Matrices: A Theoretical and Computational Framework with Applications in Imaging," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 1, pp. 148–172, Jan. 2013.

[72] H. Kasai, "Online Low-Rank Tensor Subspace Tracking from Incomplete Data by Cp Decomposition Using Recursive Least Squares," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Mar. 2016.

[73] U. Kang, E. Papalexakis, A. Harpale, and C. Faloutsos, "GigaTensor: Scaling Tensor Analysis up by 100 Times - Algorithms and Discoveries," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2012.

[74] Q. Song, H. Ge, J. Caverlee, and X. Hu, "Tensor Completion Algorithms in Big Data Analytics," *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 1, pp. 1–48, Jan. 2019.

[75] Y. Zhang, J. Xiao, and M. Shah, "Motion Layer Based Object Removal in Videos," in *IEEE Workshops on Applications of Computer Vision*, vol. 1, Jan. 2005, pp. 516–521.

[76] J. Jia, W. Tai-Pang, Y.-W. Tai, and C.-K. Tang, "Video Repairing: Inference of Foreground and Background Under Severe Occlusion," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, Jun. 2004.

[77] K. A. Patwardhan, G. Sapiro, and M. Bertalmio, "Video Inpainting Under Constrained Camera Motion," *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 545–553, Feb. 2007.

[78] Y.-T. Jia, S.-M. Hu, and R. R. Martin, "Video Completion Using Tracking and Fragment Merging," *The Visual Computer*, vol. 21, no. 8, pp. 601–610, Sep. 2005.

[79] M. Granados, K. I. Kim, J. Tompkin, J. Kautz, and C. Theobalt, "Background Inpainting for Videos with Dynamic Objects and a Free-Moving Camera," in *European Conference on Computer Vision*, 2012, pp. 682–695.

[80] M. Ebdelli, O. Le Meur, and C. Guillemot, "Video Inpainting with Short-Term Windows: Application to Object Removal and Error Concealment," *IEEE Transactions on Image Processing*, vol. 24, no. 10, pp. 3034–3047, Oct. 2015.

[81] P. Pérez, M. Gangnet, and A. Blake, "Poisson Image Editing," in *ACM SIGGRAPH*, Jul. 2003.

[82] V. Cheung, B. J. Frey, and N. Jojic, "Video Epitomes," *International Journal of Computer Vision*, vol. 76, no. 2, pp. 141–152, Feb. 2008.

[83] Y. Wexler, E. Shechtman, and M. Irani, "Space-Time Completion of Video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 3, pp. 463–476, 2007.

[84] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing," *ACM Transactions on Graphics*, vol. 28, no. 3, 24:1–24:11, Jul. 2009.

[85] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention*, 2015.

[86] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," in *European Conference on Computer Vision*, 2016.

[87] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2016.

[88] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, Jul. 2017.

[89] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Neural Information Processing Systems*, Dec. 2014.

[90] X. Sun, R. Szeto, and J. J. Corso, "A Temporally-Aware Interpolation Network for Video Frame Inpainting," in *Asian Conference on Computer Vision*, Dec. 2018.

[91] R. Szeto, X. Sun, K. Lu, and J. J. Corso, "A Temporally-Aware Interpolation Network for Video Frame Inpainting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, May 2020.

[92] S. Niklaus, L. Mai, and F. Liu, "Video Frame Interpolation via Adaptive Separable Convolution," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 261–270.

[93] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee, "Decomposing Motion and Content for Natural Video Sequence Prediction," in *International Conference on Learning Representations*, 2017.

[94] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations*, 2015.

[95] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, "Convolutional LSTM Network: A Machine Learning Approach For Precipitation Nowcasting," in *Neural Information Processing Systems*, 2015, pp. 802–810.

[96] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[97] M. Mathieu, C. Couprie, and Y. LeCun, "Deep Multi-Scale Video Prediction Beyond Mean Square Error," *International Conference on Learning Representations*, 2016.

[98] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[99] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral Normalization for Generative Adversarial Networks," in *International Conference on Learning Representations*, Feb. 2018.

[100] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing Human Actions: A Local SVM Approach," in *International Conference on Pattern Recognition*, vol. 3, 2004, pp. 32–36.

[101] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A Dataset Of 101 Human Actions Classes From Videos In The Wild," UCF Center for Research in Computer Vision, Technical report, 2012. [Online]. Available: `https://www.crcv.ucf.edu/papers/UCF101_CRCV-TR-12-01.pdf`.

[102] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A Large Video Database For Human Motion Recognition," in *IEEE International Conference on Computer Vision*, 2011, pp. 2556–2563.

[103] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, Berg, Alexander C., and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[104] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic Differentiation in PyTorch," in *Neural Information Processing Systems (Workshop Track)*, 2017.

[105] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore,

Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems,” Google, White paper, 2015, p. 19. [Online]. Available: `https://www.tensorflow.org/`.

[106] D. P. Kingma and J. L. Ba, “Adam: A Method For Stochastic Optimization,” in *International Conference on Learning Representations*, 2015.

[107] X. Glorot and Y. Bengio, “Understanding The Difficulty Of Training Deep Feedforward Neural Networks,” in *International Conference on Artificial Intelligence and Statistics*, 2010.

[108] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, “Scikit-image: Image processing in Python,” *PeerJ*, vol. 2, e453, 2014.

[109] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, “Generative Image Inpainting with Contextual Attention,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2018.

[110] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-Image Translation with Conditional Adversarial Networks,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Jul. 2017.

[111] X. Yang, D. Xie, and X. Wang, “Crossing-Domain Generative Adversarial Networks for Unsupervised Multi-Domain Image-to-Image Translation,” in *ACM Multimedia*, Oct. 2018.

[112] A. Bansal, S. Ma, D. Ramanan, and Y. Sheikh, “Recycle-GAN: Unsupervised Video Retargeting,” in *European Conference on Computer Vision*, 2018.

[113] K. Park, S. Woo, D. Kim, D. Cho, and I. S. Kweon, “Preserving Semantic and Temporal Consistency for Unpaired Video-to-Video Translation,” in *ACM Multimedia*, Oct. 2019.

[114] Y. Chen, Y. Pan, T. Yao, X. Tian, and T. Mei, “Mocycle-GAN: Unpaired Video-to-Video Translation,” in *ACM Multimedia*, Oct. 2019.

[115] R. Zhang, P. Isola, and A. A. Efros, “Colorful Image Colorization,” in *European Conference on Computer Vision*, 2016.

[116] R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros, “Real-Time User-Guided Image Colorization with Learned Deep Priors,” *ACM Transactions on Graphics*, vol. 36, no. 4, 119:1–119:11, Jul. 2017.

[117] C. Dong, C. C. Loy, K. He, and X. Tang, “Image Super-Resolution Using Deep Convolutional Networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, Feb. 2016.

[118] X. Hong, P. Xiong, R. Ji, and H. Fan, “Deep Fusion Network for Image Completion,” in *ACM Multimedia*, Oct. 2019.

[119] N. Bonneel, J. Tompkin, K. Sunkavalli, D. Sun, S. Paris, and H. Pfister, “Blind Video Temporal Consistency,” *ACM Transactions on Graphics*, vol. 34, no. 6, 196:1–196:9, Oct. 2015.

[120] X. Dong, B. Bonev, Y. Zhu, and A. L. Yuille, “Region-Based Temporally Consistent Video Post-Processing,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2015.

[121] C.-H. Yao, C.-Y. Chang, and S.-Y. Chien, “Occlusion-Aware Video Temporal Consistency,” in *ACM Multimedia*, Oct. 2017.

[122] R. Szeto, M. El-Khamy, J. Lee, and J. J. Corso, “HyperCon: Image-To-Video Model Transfer for Video-To-Video Translation Tasks,” in *IEEE Winter Conference on Applications of Computer Vision*, Jan. 2021.

[123] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, “StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2018.

[124] M.-Y. Liu, T. Breuel, and J. Kautz, “Unsupervised Image-to-Image Translation Networks,” in *Neural Information Processing Systems*, Dec. 2017.

[125] Z. Yi, H. Zhang, P. Tan, and M. Gong, “DualGAN: Unsupervised Dual Learning for Image-to-Image Translation,” in *IEEE International Conference on Computer Vision*, Oct. 2017.

[126] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks,” in *IEEE International Conference on Computer Vision*, Oct. 2017.

[127] M. Mirza and S. Osindero, “Conditional Generative Adversarial Nets,” *arXiv preprint arXiv:1411.1784*, Nov. 2014.

[128] X. Wei, J. Zhu, S. Feng, and H. Su, “Video-to-Video Translation with Global Temporal Consistency,” in *ACM Multimedia*, Oct. 2018.

[129] L. Shen, R. Hong, H. Zhang, H. Zhang, and M. Wang, “Single-Shot Semantic Image Inpainting with Densely Connected Generative Networks,” in *ACM Multimedia*, Oct. 2019.

[130] Z. Guo, Z. Chen, T. Yu, J. Chen, and S. Liu, "Progressive Image Inpainting with Full-Resolution Residual Network," in *ACM Multimedia*, Oct. 2019.

[131] F. A. Reda, G. Liu, K. J. Shih, R. Kirby, J. Barker, D. Tarjan, A. Tao, and B. Catanzaro, "SDC-Net: Video Prediction Using Spatially-Displaced Convolution," in *European Conference on Computer Vision*, 2018.

[132] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles, "ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding," in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2015.

[133] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved Training of Wasserstein GANs," in *Neural Information Processing Systems*, Dec. 2017.

[134] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and Checkerboard Artifacts," *Distill*, vol. 1, no. 10, Oct. 2016. [Online]. Available: `http://distill.pub/2016/deconv-checkerboard`.

[135] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-Scale Video Classification With Convolutional Neural Networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[136] E. Santana and G. Hotz, "Learning a Driving Simulator," *arXiv preprint arXiv:1608.01230*, 2016.

[137] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied To Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[138] *FaceGen — 3D Faces and Heads*. [Online]. Available: `https://facegen.com/` (visited on 05/27/2020).

[139] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised Learning of Video Representations Using LSTMs," in *International Conference on Machine Learning*, 2015, pp. 843–852.

[140] R. Szeto, S. Stent, G. Ros, and J. J. Corso, "A Dataset To Evaluate The Representations Learned By Video Prediction Models," in *International Conference on Learning Representations (Workshop Track)*, Apr. 2018.

[141] V. Pătrăucean, A. Handa, and R. Cipolla, "Spatio-Temporal Video Autoencoder with Differentiable Memory," in *International Conference on Learning Representations (Workshop Track)*, May 2016.

[142] N. Kalchbrenner, A. v. d. Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu, “Video Pixel Networks,” in *International Conference on Machine Learning*, 2017.

[143] E. Denton and V. Birodkar, “Unsupervised Learning of Disentangled Representations from Video,” in *Neural Information Processing Systems*, Dec. 2017.

[144] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, “Human-Level Concept Learning Through Probabilistic Program Induction,” *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[145] A. Krizhevsky and G. Hinton, “Learning Multiple Layers of Features from Tiny Images,” University of Toronto, Technical report, 2009. [Online]. Available: `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

[146] C. Yeager, *Everything You Need to Know About Chroma Key and Green Screen Footage*, Jul. 2019. [Online]. Available: `https://www.premiumbeat.com/blog/chroma-key-green-screen-guide/` (visited on 06/07/2020).

[147] *Production Notes: Match Moving — Nevada Film Office*, Feb. 2018. [Online]. Available: `https://nevadafilm.com/production-notes-match-moving/` (visited on 06/07/2020).

[148] C. Gao, A. Saraf, J.-B. Huang, and J. Kopf, “Flow-Edge Guided Video Completion,” in *European Conference on Computer Vision*, 2020.

[149] J. Shen, S. Zafeiriou, G. G. Chrysos, J. Kossaifi, G. Tzimiropoulos, and M. Pantic, “The First Facial Landmark Tracking In-the-Wild Challenge: Benchmark and Results,” in *IEEE International Conference on Computer Vision Workshops*, Dec. 2015.

[150] R. Szeto and J. J. Corso, “The DEVIL is in the Details: A Diagnostic Evaluation Benchmark for Video Inpainting,” *arXiv preprint arXiv:2105.05332*, May 2021.

[151] *Flickr*, https://www.flickr.com/. [Online]. Available: `https://www.flickr.com/` (visited on 03/04/2021).

[152] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common Objects in Context,” in *European Conference on Computer Vision*, 2014.

[153] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[154] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, *Detectron2*, 2019. [Online]. Available: `https://github.com/facebookresearch/detectron2`.

[155] D. A. Forsyth, J. Malik, M. M. Fleck, H. Greenspan, T. Leung, S. Belongie, C. Carson, and C. Bregler, “Finding Pictures of Objects in Large Collections of Images,” in *International Workshop on Object Representation in Computer Vision*, 1996.

[156] M. A. Fischler and R. C. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[157] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded Up Robust Features,” in *European Conference on Computer Vision*, 2006.

[158] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Neural Information Processing Systems*, 2012.

[159] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is All You Need,” in *Neural Information Processing Systems*, 2017.