Modeling Structured Dynamics with Deep Neural Networks

by

Ruben Villegas

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
2019

Ruben Villegas

rubville@umich.edu

ORCID iD: 0000-0001-8488-9158

ACKNOWLEDGEMENTS

I would first like to thank God for guiding me through this road from not knowing what my future will be after basketball was gone; all the way to have the opportunity to interact with and befriend some of the smartest people on the planet. I would also like to thank my parents for raising me with the belief that I have to *earn my spot* in anything I do, my brothers for always challenging me and inspiring me, and my amazing wife for keeping sane and supporting me through this process. My grandmothers, aunts, uncles and cousins for being there when I needed them the most, and my grandfathers for watching over me. All the people that guided me and helped me through my academic path, there are too many to list, but I will highlight a few:

My undergraduate research adviser, Mubarak Shah, for believing in me and giving me the opportunity to be involved in his research lab. Brian Moore and Niels Da Vitoria Lobo for interviewing me for the scholarship that helped get involved in research early on. Afshin Dehghan, Amir R. Zamir and Enrique Ortiz for their research advice throughout my undergrad and for continuing to challenge me on making algorithms that work outside of simplified datasets. Omar Orefeij, Shayan Modiri, Mahdi Kalayeh, Gonzalo Vaca and others at CRCV for their friendship and advice. Lawrence Eric Meyer, my introduction to programming in C++ teacher at Miami Dade College, for giving me the initial tool that has helped me throughout my career in Computer Science.

Jimei Yang, Duygu Ceylan, Dumitru Erhan, Quoc V. Le, Arkanath Pathak, Harini Kannan, Nevan Wichers, Alejandro Troccoli, Iuri Frosio, Wonmin Byeon, Stephen Tyree and many others for their advice, collaborations, and helping shape my thinking as a researcher. I am grateful for having the opportunity to work on research projects at Adobe Research, Google Brain and NVIDIA Research allowed me to have first hand experience of a wide variety of problems in artificial intelligence.

My adviser and dissertation Committee Chair Honglak Lee, who I have the privilege to work with since the start of my PhD, and briefly work with during the Summer of 2013 thanks to Michigan's SROP. He played a very important role in my life as a graduate student; from rooting for me to being accepted in Michigan's PhD program; to inspiring me to always accomplish more as a PhD student; to guiding me on the many projects we worked on; to believing in me throughout my PhD.

Jia Deng, Chad Jenkins and Jason Corso who kindly served as my dissertation committee, and provided valuable feedback on this dissertation.

All my amazing lab mates and collaborators for fruitful discussions and collaborations: Yuting Zhang, Seunghoon Hong, Xinchen Yan, Junhyuk Oh, Ye Liu, Lajanugen Logeswaran, Kihyuk Sohn, Scott Reed, Hongji Wang, Michael Chang, Sungryull Sohn, Kibok Lee, Jongwook Choi, Yijie Guo, Wenling Shang, Yunseok Yang, Wilka Carvalho, Tianchen Zhao, and Kimin Lee.

Last but not least, I would like to thank Rackham Merit Fellowship and the University of Michigan for supporting me throughout my PhD, and for introducing me to other PhD students in areas outside of my own.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

ABSTRACT

Deep neural networks have become powerful machinery for identifying important patterns from raw input data when large amounts of data is available. Research adopting neural networks has excelled in tasks such as object recognition, reinforcement learning, speech recognition, image in-painting, sequence modeling, amongst others. Previous works in computer vision have notably excelled at inferring information about the input data; either from sequence of frames or single frames. However, very few works have focused on modeling structured motion dynamics for generative tasks. Structured motion is defined as the constant topological configuration of objects maintained through time in sequential data. In this thesis, I develop new neural networks that effectively model structured motion dynamics useful for generative tasks such as future motion prediction and transfer. Accurate structured dynamic models are an important piece in achieving general artificial intelligence. It has been recently shown that agents equipped with such models can learn from environments with far less interactions due to being able to predict the consequences of their actions in the environment. Additionally, accurate motion dynamic models are be useful for applications such as motion editing, motion transfer, and others. Such applications can enhance visual artists ability to create content for the web or can assist movie makers when transferring motion from actors into movie characters.

This thesis initially presents motion dynamics models in two dimensions: I first present a neural network architecture that decomposes video into two information pathways that deal with video dynamics and frame spatial layout separately. The two pathways are later combined to generate future frames that contain highly structured objects moving. Second, I propose to take it a step further by having a motion stream that is visually interpretable. Specifically, there is a motion stream that predicts structured motion dynamics as landmarks of the moving structures that evolve through time, and there is an image generation module that generates future frames given the landmarks and a single frame from the past using image analogy principles. Next, we keep the image analogy principles of our previous work, however, we formulate the video prediction problem such that general features for moving objects structures are learned. Finally, by taking advantage of recent advances in computational devices for large scale deep learning research, I present a study on the effects of maximal capacity and minimal inductive bias of neural networks based video prediction frameworks. From our very thorough evaluation and experimentation, we find that network capacity plays a very important role in the performance of deep networks for video prediction that can be applied to any of the previously investigated methods.

Consequently, this thesis presents motion dynamics models in three dimensions: I propose a neural kinematics network with adversarial cycle consistency. Specifically, I propose a layer based on the kinematic equations that takes advantage of the backpropagation algorithm used to optimize neural networks to automatically discover rotation angles that represent pure motion which can be used for motion transfer from one kinematic structure into another. Because of the unsupervised nature of learning, the learned model generalizes to never before seen human video from which motion data is extracted using an off-the-shelf algorithm.

Overall, this thesis focuses on modeling structured dynamics using the representational power of deep neural networks. Modeling structured dynamics is an important problem in both general artificial intelligence, as well as, in applications dealing video editing, video generation, video understanding and animation.

CHAPTER I

Introduction

Deep neural networks have served as proxy to make great progress in computer vision and machine learning in the last few years. Increase in training data and computational power have enabled us to harvest the pattern recognition capabilities of deep neural as data becomes more abundant. Traditionally, deep neural networks have excelled at recognizing patterns from high-dimensional data for tasks such as object, activity, and speech recognition (Krizhevsky et al., 2012; Szegedy et al., 2015; Graves et al., 2013). Recently, deep neural networks have been used for problems that require inferring information that depends on clearly defined object structures from raw high-dimensional inputs; such as image generation, object segmentation and pose estimation (Goodfellow et al., 2014a; Hong et al., 2015; Newell et al., 2016). However, modeling structured dynamics from sequential perceptual data has not been extensively studied. We define structures as the constant topological configuration of objects that can perceptually change based on the object state or point of view. Therefore, modeling structured dynamics is defined as the ability of models to understand and maintain the topological configuration of the objects being observed. In this thesis, I focus on modeling the challenging structured dynamics observed in video data, and I propose multiple perspectives and approaches ranging from end-to-end methods, to interpretable hierarchical models, to large-scale models across multiple devices.

Accurate dynamic models are important to predict what will happen next. Having an agent that can imagine the consequence of its actions, as well as, how its surrounding environment will behave is very important for achieving general artificial intelligence. Model-based reinforcement learning from pixels has shown great progress in recent works. Particularly, future prediction models combined with simple planning algorithms (Hafner et al., 2019) or used for policy learning Kaiser et al. (2019) have been shown to perform equally or better than model-free methods with far less interactions with the environment. Additionally, Ebert et al. (2018) showed that future prediction methods are also useful for robotic control where the reward can be defined telling a robot to move a specific pixel or region of pixels to a goal location. The predicted future frames can then be used to determine if the goal has been achieved.

We initially focus on modeling dynamics in two dimensional space. In Chapter II I propose a neural network architecture that decomposes motion and frame content into two streams of information that are later combined to predict the next frame. I provide visual evidence that our method identifies and only moves the moving structures in the video by computing flow vectors in the predicted frames and comparing against the ground truth. In Chapter III, I explore an alternative but complementary way for modeling video dynamics which we also use for future frame prediction. Instead of a simple information separation of motion and content, the motion stream is modeled as a sequence of landmarks that represent the high-level structure of the objects moving in the video. The predicted future landmarks are later used to generate all future video frames given a single frame from the past. This approach enables future frame prediction for more than an order of magnitude of time steps into the future in comparison to previous methods. In Chapter IV, we explore a method inspired by Chapter III in which general representation of the moving structures is learned directly from input frames rather than using predefined landmark representations. Finally, in Chapter V, I explore the advantages of large scale training for generating future video frames. I present a study that maximizes the capacity of neural network architectures while minimizing inductive bias in order to achieve high quality future frame prediction of videos with highly structured motion. The findings discussed in Chapter V are applicable to all previous works presented in this thesis. Modeling structured dynamics in two dimensions is useful, however, our world lives in a three dimensional plane. Therefore, simply modeling two dimension can result in inaccuracies due to occlusions caused by the missing third dimension.

Subsequently, I move on to modeling structured dynamics in three dimensional space. Three dimensional information includes height, width and depth information which is necessary to more accurate model the world. For example, if I need to model the precise location of an object that is moving directly towards me, I will need to know depth information of that object which is not easy to infer from two dimensional input. This level accuracy is necessary for applications such as autonomous driving, navigation, object tracking, amongst others; where high accuracy of the location of an agent and its surroundings is extremely important.

In Chapter VI, I propose a new neural network with a forward kinematics layer and adversarial cycle consistency training technique for unsupervised motion retargeting in three dimensions. I demonstrate that our method is able to perform online motion retargeting by modeling pure motion features that are independent of the kinematic structures performing the motion. In essence, our method automatically discovers the high-level structured motion dynamics as rotation angles to achieve a desired motion sequence given a reference motion. Additionally, I show evidence that the solution learned by our method is general and can be used to retarget motion from never before seen human videos.

In a nutshell, this thesis investigates neural network methods for modeling structured dynamics. Accurate dynamic models should capture general high-level motion dynamics features within their transition functions such that extrapolation, interpolation and transfer is possible. In addition, such models contribute to the progress in many applications ranging from model-based reinforcement learning, to robotics, to generative models of video, to character animation.

1.1 Summary of Research Contributions

Decomposing Motion and Content for Natural Video Prediction (Chapter II)

This chapter tackles the high-dimensional video prediction problem by separating the temporal and spatial sources of information present in videos. Videos contain many factors of variation present through time such as color changes, pixel displacement, previously unseen pixel appearance, and others. This paper proposes a new neural network architecture that consists of an encoder network that observes pixel differences through time, another network that observes pixel layout in the last frame, and a final network that combines the outputs of the two encoders. I show the effectiveness of the proposed neural network architecture, and outperform previous state-of-the-art methods in video prediction.

Learning to Generate Long-term Future via Hierarchical Prediction (Chapter III)

This chapter takes a step forward towards a solution to the video prediction problem by predicting future frames in a hierarchical fashion. A notorious problem in pixel-level video prediction is that frames exponentially degrade as I recursively predict solely pixels deeper into the future. I mitigate this problem by making predictions of the future in high-level feature space (i.e., object landmarks), and then use the predicted high-level features to generate all future frames given a single input frame from the past. I show that our method can predict an order of magnitude more frames compared to the previous state-of-the-art.

Hierarchical Long-term Video Prediction without Supervision (Chapter IV)

This chapter focuses on a more general version of the hierarchical method in Chapter III. We formulate the frame prediction process to discover general learned features rather than predefined landmarks. Deep analogy making is used as the driving force to discover features that represent the moving objects structures. We show that our method is capable of generating long term video without the need of predefined landmark features.

High Fidelity Video Prediction with Large Neural Nets (Chapter V)

This chapter takes a slightly different but complementary route to previous chapters on modeling structured dynamics with neural networks. Rather than building additional specialized layers or losses, I focus on minimizing inductive bias and maximizing the capacity of deep neural networks to the limits of current computational devices. I show that state-of-the-art future frame prediction is achievable in multiple datasets with different challenges (object interaction, structured motion and moving background) using multiple evaluation metrics (per-frame and full-sequence based evaluations).

Neural Kinematic Networks for Unsupervised Motion Retargeting (Chapter VI)

This chapter focuses on the motion retargeting between 3D humanoid characters, an important problem in robotics and character animation. I mainly focus on the motion retargeting scenario when ground truth input/target motion pairs are not available for learning a retargeting model. In this work, I propose a novel neural network architecture with a differentiable forward kinematics layer and an adversarial cycle consistency training for unsupervised motion retargeting. In our experiments, I show that our network is able to successfully perform motion retargeting between 3D characters with only a simple objective during training. Additionally, I show that our network can retarget motions from estimated 3D human pose motion sequence into 3D characters without ever training on estimated human pose or the human skeleton used by the pose estimation algorithm.

CHAPTER II

Decomposing Motion and Content for Natural Video Prediction

We propose a deep neural network for the prediction of future frames in natural video sequences. To effectively handle complex evolution of pixels in videos, we propose to decompose the motion and content, two key components generating dynamics in videos. Our model is built upon the Encoder-Decoder Convolutional Neural Network and Convolutional LSTM for pixel-level prediction, which independently capture the spatial layout of an image and the corresponding temporal dynamics. By independently modeling motion and content, predicting the next frame reduces to converting the extracted content features into the next frame content by the identified motion features, which simplifies the task of prediction. Our model is end-to-end trainable over multiple time steps, and naturally learns to decompose motion and content without separate training. We evaluate the proposed network architecture on human activity videos using KTH, Weizmann action, and UCF-101 datasets. We show state-of-the-art performance in comparison to recent approaches. To the best of our knowledge, this is the first end-to-end trainable network architecture with motion and content separation to model the spatio-temporal dynamics for pixel-level future prediction in natural videos.

2.1 Introduction

Understanding videos has been one of the most important tasks in the field of computer vision. Compared to still images, the temporal component of videos provides much richer descriptions of the visual world, such as interaction between objects, human activities, and so on. Amongst the various tasks applicable on videos, the task of anticipating the future has recently received increased attention in the research community. Most prior works in this direction focus on predicting high-level semantics in a video such as action (Vondrick et al., 2015; Ryoo, 2011; Lan et al., 2014), event (Yuen and Torralba, 2010a; Hoai and Torre, 2013) and motion (Pintea et al., 2014; Walker et al., 2014; Pickup et al., 2014; Walker et al., 2016). Forecasting semantics provides information about *what will happen* in a video, and is essential to automate decision making. However, the predicted semantics are often specific to a particular task and provide only a partial description of the future. Also, training such models often requires heavily labeled training data which leads to tremendous annotation costs especially with videos.

In this work, we aim to address the problem of prediction of future frames in natural video sequences. Pixel-level predictions provide dense and direct description of the visual world, and existing video recognition models can be adopted on top of the predicted frames to infer various semantics of the future. Spatio-temporal correlations in videos provide a self-supervision for frame prediction, which enables purely unsupervised training of a model by observing raw video frames. Unfortunately, estimating frames is an extremely challenging task; not only because of the inherent uncertainty of the future, but also various factors of variation in videos leading to complicated dynamics in raw pixel values. There have been a number of recent attempts on frame prediction (Srivastava et al., 2015; Mathieu et al., 2015; Oh et al., 2015; Goroshin et al., 2015; Lotter et al., 2015; Ranzato et al., 2014), which use a single encoder that needs to reason about all the different variations occurring in videos in order to make predictions of the future, or infer extra information like foreground segmentation masks and static background (Vondrick et al., 2016).

We propose a Motion-Content Network (MCnet) for robust future frame prediction. Our intuition is to split the inputs for video prediction into two easily identifiable groups, motion and content, and independently capture each information stream with separate encoder pathways. In this architecture, the *motion* pathway encodes the local dynamics of spatial regions, while the *content* pathway encodes the spatial layout of the salient parts of an image. The prediction of the future frame is then achieved by transforming the content of the last observed frame given the identified dynamics up to the last observation. Somewhat surprisingly, we show that such a network is end-to-end trainable *without individual path way supervision*. Specifically, we show that an asymmetric architecture for the two pathways enables such decompositions without explicit supervision. The contributions of this paper are summarized below:

- We propose MCnet for the task of frame prediction, which separates the information streams (motion and content) into different encoder pathways.
- The proposed network is end-to-end trainable and naturally learns to decompose motion and content without separate training, and reduces the task of frame prediction to transforming the last observed frame into the next by the observed motion.
- We evaluate the proposed model on challenging real-world video datasets, and show that it outperforms previous approaches on frame prediction.

The rest of the paper is organized as follows. We briefly review related work in Section 2.2, and introduce an overview of the proposed algorithm in Section 2.3. The detailed configuration of the proposed network is described in Section 2.4. Section 2.5 describes training and inference procedure. Section 2.6 illustrates implementation details and experimental results on challenging benchmarks.

2.2 Related work

The problem of visual future prediction has received growing interests in the computer vision community. It has led to various tasks depending on the objective of future prediction, such as human activity (Vondrick et al., 2015; Ryoo, 2011; Lan et al., 2014), event (Yuen and Torralba, 2010a; Hoai and Torre, 2013) and geometric path (Walker et al., 2014). Although previous work achieved reasonable success in specific tasks, they are often limited to estimating predefined semantics, and require fully-labeled training data. To alleviate this issue, approaches predicting representation of the future beyond semantic labels have been proposed. Walker et al. (2014) proposed a data-driven approach to predict the motion of a moving object, and coarse hallucination of the predicted motion. Vondrick et al. (2015) proposed a deep regression network to predict feature representations of the future frames. These approaches are supervised and provide coarse predictions of how the future will look like. Our work also focuses on unsupervised learning for prediction of the future, but to a more direct visual prediction task: frame prediction.

Compared to predicting semantics, pixel-level prediction has been less investigated due to the difficulties in modeling evolution of raw pixels over time. Fortunately, recent advances in deep learning provide a powerful tool for sequence modeling, and enable the creation of novel architectures for modeling complex sequential data. Ranzato et al. (2014) applied a recurrent neural network developed for language modeling to frame prediction by posing the task as classification of each image region to one of quantized patch dictionaries. Srivastava et al. (2015) applied a sequence-to-sequence model to video prediction, and showed that Long Short-Term Memory (LSTM) is able to capture pixel dynamics. Oh et al. (2015) proposed an action-conditional encoder-decoder network to predict future frames in Atari games. In addition to the different choices of architecture, some other works addressed the importance of selecting right objective function: Lotter et al. (2015) used adversarial loss with combined CNN and LSTM architectures, and Mathieu et al. (2015)

employed similar adversarial loss with additional regularization using a multi-scale encoder-decoder network. Finn et al. (2016b) constructed a network that predicts transformations on the input pixels for next frame prediction. Patraucean et al. (2015) proposed a network that by explicitly predicting optical flow features is able to predict the next frame in a video. Vondrick et al. (2016) proposed a generative adversarial network for video which, by generating a background-foreground mask, is able to generate realistic-looking video sequences. However, none of the previously mentioned approaches exploit spatial and temporal information separately in an unsupervised fashion. In terms of the way data is observed, the closest work to ours is Xue et al. (2016). The differences are (1) Our model is deterministic and theirs is probabilistic, (2) our motion encoder is based on convolutional LSTM (Shi et al., 2015) which is a more natural module to model long-term dynamics, (3) our content encoder observes a single scale input and theirs observes many scales, and (4) we directly generate image pixels values, which is a more complicated task. We aim to exploit the existing spatio-temporal correlations in videos by decomposing the motion and content in our network architecture.

To the best of our knowledge, the idea of separating motion and content has not been investigated in the task of unsupervised deterministic frame prediction. The proposed architecture shares similarities to the two-stream CNN (Simonyan and Zisserman, 2014), which is designed for action recognition to jointly exploit the information from frames and their temporal dynamics. However, in contrast to their network we aim to learn features for temporal dynamics directly from the raw pixels, and we use the identified features from the motion in combination with spatial features to make pixel-level predictions of the future.

2.3 Algorithm Overview

In this section, we formally define the task of frame prediction and the role of each component in the proposed architecture. Let $\mathbf{x}_t \in \mathrm{R}^{w \times h \times c}$ denote the t-th frame in an input video $\mathbf{x}$, where w, h, and c denote width, height, and number of channels, respectively. The objective of frame prediction is to generate the future frame $\hat{\mathbf{x}}_{t+1}$ given the input frames $\mathbf{x}_{1:t}$.

At the t-th time step, our network observes a history of previous consecutive frames up to frame t, and generates the prediction of the next frame $\hat{\mathbf{x}}_{t+1}$ as follows:

- **Motion Encoder** recurrently takes an image difference input between frame $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$ starting from $t = 2$, and produces the hidden representation $\mathbf{d}_t$ encoding the temporal dynamics of the scene components (Section 2.4.1).

- **Content Encoder** takes the last observed frame $\mathbf{x}_t$ as an input, and outputs the hidden representation $\mathbf{s}_t$ that encodes the spatial layout of the scene (Section 2.4.2).

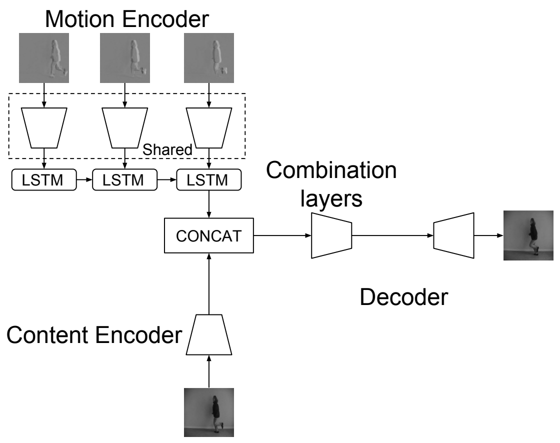
- **Multi-Scale Motion-Content Residual** takes the computed features, from both the motion and content encoders, at every scale right before pooling and computes residuals $\mathbf{r}_t$ (He et al., 2015) to aid the information loss caused by pooling in the encoding phase (Section 2.4.3).

- **Combination Layers and Decoder** takes the outputs from both encoder pathways and residual connections, $\mathbf{d}_t$, $\mathbf{s}_t$, and $\mathbf{r}_t$, and combines them to produce a pixel-level prediction of the next frame $\hat{\mathbf{x}}_{t+1}$ (Section 2.4.4).

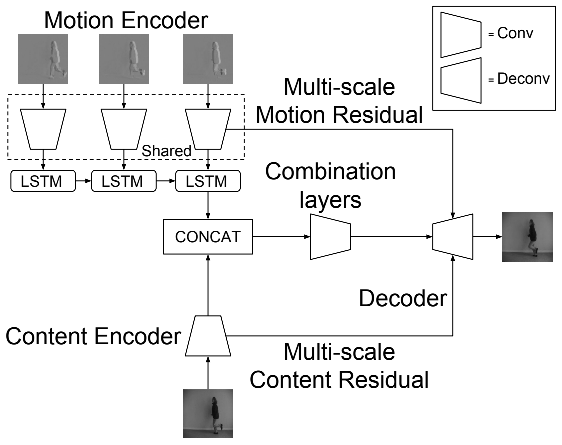
The overall architecture of the proposed algorithm is described in Figure 2.1. The prediction of multiple frames, $\hat{\mathbf{x}}_{t+1:t+T}$, can be achieved by recursively performing the above procedures over T time steps (Section 2.5). Each component in the proposed architecture is described in the following section.

2.4 Architecture

This section describes the detailed configuration of the proposed architecture, including the two encoder pathways, multi-scale residual connections, combination layers, and decoder.



(a) Base MCnet



(b) MCnet with Multi-scale Motion-Content Residuals

Figure 2.1: Overall architecture of the proposed network. (a) illustrates MCnet without the Motion-Content Residual *skip connections*, and (b) illustrates MCnet with such connections. Our network observes a history of image differences through the motion encoder and last observed image through the content encoder. Subsequently, our network proceeds to compute motion-content features and communicates them to the decoder for the prediction of the next frame.

2.4.1 Motion Encoder

The motion encoder captures the temporal dynamics of the scene's components by recurrently observing subsequent difference images computed from $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$, and outputs motion features by

$$[\mathbf{d}_t, \mathbf{c}_t] = f^{\text{dyn}}\left(\mathbf{x}_t - \mathbf{x}_{t-1}, \mathbf{d}_{t-1}, \mathbf{c}_{t-1}\right), \tag{2.1}$$

where $\mathbf{x}_t - \mathbf{x}_{t-1}$ denotes element-wise subtraction between frames at time t and $t-1$, $\mathbf{d}_t \in \mathbb{R}^{w' \times h' \times c'}$ is the feature tensor encoding the motion across the observed difference image inputs, and $\mathbf{c}_t \in \mathbb{R}^{w' \times h' \times c'}$ is a memory cell that retains information of the dynamics observed through time. f^{dyn} is implemented in a fully-convolutional way to allow our model to identify local dynamics of frames rather than complicated global motion. For this, we use an encoder CNN with a Convolutional LSTM (Shi et al., 2015) layer on top.

2.4.2 Content Encoder

The content encoder extracts important spatial features from a single frame, such as the spatial layout of the scene and salient objects in a video. Specifically, it takes the last observed frame $\mathbf{x}_t$ as an input, and produces content features by

$$\mathbf{s}_t = f^{\text{cont}}\left(\mathbf{x}_t\right), \tag{2.2}$$

where $\mathbf{s}_t \in \mathbb{R}^{w' \times h' \times c'}$ is the feature encoding the spatial content of the last observed frame, and f^{cont} is implemented by a Convolutional Neural Network (CNN) that specializes on extracting features from single frame.

It is important to note that our model employs an *asymmetric* architecture for the motion and content encoder. The content encoder takes the last observed frame, which keeps the most critical clue to reconstruct spatial layout of near future, but has no information about dynamics. On the other hand, the motion encoder takes a history of previous image differences, which are less informative about the future spatial layout compared to the last observed frame, yet contain important spatio-temporal variations occurring over time. This asymmetric architecture encourages encoders to exploit each of two pieces of critical information to predict the future content and motion individually, and enables the model to learn motion and content decomposition naturally without any supervision.

2.4.3 Multi-scale Motion-Content Residual

To prevent information loss after the pooling operations in our motion and content encoders, we use residual connections (He et al., 2015). The residual connections in our network communicate motion-content features at every scale into the decoder layers after unpooling operations. The residual feature at layer l is computed by

$$\mathbf{r}_t^l = f^{\text{res}}\left(\left[\mathbf{s}_t^l, \mathbf{d}_t^l\right]\right)^l, \tag{2.3}$$

where $\mathbf{r}_t^l$ is the residual output at layer l, $\left[\mathbf{s}_t^l, \mathbf{d}_t^l\right]$ is the concatenation of the motion and content features along the depth dimension at layer l of their respective encoders, $f^{\text{res}}\left(.\right)^l$ the residual function at layer l implemented as consecutive convolution layers and rectification with a final linear layer.

2.4.4 Combination Layers and Decoder

The outputs from the two encoder pathways, $\mathbf{d}_t$ and $\mathbf{s}_t$, encode a high-level representation of motion and content, respectively. Given these representations, the objective of the decoder is to generate a pixel-level prediction of the next frame $\hat{\mathbf{x}}_{t+1} \in \mathbb{R}^{w \times h \times c}$. To this end, it first combines the motion and content back into a unified representation by

$$\mathbf{f}_t = g^{\text{comb}}\left([\mathbf{d}_t, \mathbf{s}_t]\right), \tag{2.4}$$

where $[\mathbf{d}_t, \mathbf{s}_t] \in \mathbb{R}^{w' \times h' \times 2c'}$ denotes the concatenation of the higher-level motion and content features in the depth dimension, and $\mathbf{f}_t \in \mathbb{R}^{w' \times h' \times c'}$ denotes the combined high-level representation of motion and content. g^{comb} is implemented by a CNN with bottleneck layers (Hinton and Salakhutdinov, 2006); it first projects both $\mathbf{d}_t$ and $\mathbf{s}_t$ into a lower-dimensional embedding space, and then puts it back to the original size to construct the combined feature $\mathbf{f}_t$. Intuitively, $\mathbf{f}_t$ can be viewed as the content feature of the next time step, $\mathbf{s}_{t+1}$, which is generated by transforming $\mathbf{s}_t$ using the observed dynamics encoded in $\mathbf{d}_t$. Then our decoder places $\mathbf{f}_t$ back into the original pixel space by

$$\hat{\mathbf{x}}_{t+1} = g^{\text{dec}}\left(\mathbf{f}_t, \mathbf{r}_t\right), \tag{2.5}$$

where $\mathbf{r}_t$ is a list containing the residual connections from every layer of the motion and content encoders before pooling sent to every layer of the decoder after unpooling. We employ the deconvolution network (Zeiler et al., 2011) for our decoder network g^{dec}, which is composed of multiple successive operations of deconvolution, rectification and unpooling with the addition of the motion-content residual connections after each unpooling operation. The output layer is passed through a $\tanh(.)$ activation function. Unpooling with fixed switches are used to upsample the intermediate activation maps.

2.5 Inference and Training

Section 2.4 describes the procedures for single frame prediction, while this section presents the extension of our algorithm for the prediction of multiple time steps.

2.5.1 Multi-step prediction

Given an input video, our network observes the first n frames as image difference between frame $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$, starting from $t = 2$ up to $t = n$, to encode initial temporal dynamics through the motion encoder. The last frame $\mathbf{x}_n$ is given to the content encoder to be transformed into the first prediction $\hat{\mathbf{x}}_{t+1}$ by the identified motion features.

For each time step $t \in [n + 1, n + T]$, where T is the desired number of prediction steps, our network takes the difference image between the first prediction $\hat{\mathbf{x}}_{t+1}$ and the previous image $\mathbf{x}_t$, and the first prediction $\hat{\mathbf{x}}_{t+1}$ itself to predict the next frame $\hat{\mathbf{x}}_{t+2}$, and so forth.

2.5.2 Training Objective

To train our network, we use an objective function composed of different sub-losses similar to Mathieu et al. (2015). Given the training data $D = \{\mathbf{x}^{(i)}_{1,\ldots,T}\}^N_{i=1}$, our model is trained to minimize the prediction loss by

$$\mathcal{L} = \alpha\mathcal{L}_{\text{img}} + \beta\mathcal{L}_{\text{GAN}}, \tag{2.6}$$

where α and β are hyper-parameters that control the effect of each sub-loss during optimization. $\mathcal{L}_{\text{img}}$ is the loss in image space from Mathieu et al. (2015) defined by

$$\mathcal{L}_{\text{img}} = \mathcal{L}_p\left(\mathbf{x}_{t+k}, \hat{\mathbf{x}}_{t+k}\right) + \mathcal{L}_{gdl}\left(\mathbf{x}_{t+k}, \hat{\mathbf{x}}_{t+k}\right), \tag{2.7}$$

$$\text{where} \quad \mathcal{L}_p(\mathbf{y},\mathbf{z}) = \sum_{k=1}^{T} ||\mathbf{y}-\mathbf{z}||_p^p, \tag{2.8}$$

$$\mathcal{L}_{gdl}(\mathbf{y},\mathbf{z}) = \sum_{i,j}^{h,w} |\left(|\mathbf{y}_{i,j}-\mathbf{y}_{i-1,j}| - |\mathbf{z}_{i,j}-\mathbf{z}_{i-1,j}|\right)|^{\lambda} + |\left(|\mathbf{y}_{i,j-1}-\mathbf{y}_{i,j}| - |\mathbf{z}_{i,j-1}-\mathbf{z}_{i,j}|\right)|^{\lambda}. \tag{2.9}$$

Here, $\mathbf{x}_{t+k}$ and $\hat{\mathbf{x}}_{t+k}$ are the target and predicted frames, respectively, and p and λ are hyper-parameters for $\mathcal{L}_p$ and $\mathcal{L}_{gdl}$, respectively. Intuitively, $\mathcal{L}_p$ guides our network to match the average pixel values directly, while $\mathcal{L}_{gdl}$ guides our network to match the gradients of such pixel values. Overall, $\mathcal{L}_{\text{img}}$ guides our network to learn parameters towards generating the correct average sequence given the input. Training to generate average sequences, however, results in somewhat blurry generations which is the reason we use an additional sub-loss. $\mathcal{L}_{\text{GAN}}$ is the generator loss in adversarial training to allow our model to predict realistic looking frames and it is defined by

$$\mathcal{L}_{\text{GAN}} = -\log D\left(\left[\mathbf{x}_{1:t}, G\left(\mathbf{x}_{1:t}\right)\right]\right), \tag{2.10}$$

where $\mathbf{x}_{1:t}$ is the concatenation of the input images, $\mathbf{x}_{t+1:t+T}$ is the concatenation of the ground-truth future images, $G\left(\mathbf{x}_{1:t}\right) = \hat{\mathbf{x}}_{t+1:t+T}$ is the concatenation of all predicted images along the depth dimension, and $D\left(.\right)$ is the discriminator in adversarial training. The discriminative loss in adversarial training is defined by

$$\mathcal{L}_{\text{disc}} = -\log D\left(\left[\mathbf{x}_{1:t}, \mathbf{x}_{t+1:t+T}\right]\right) - \log\left(1 - D\left(\left[\mathbf{x}_{1:t}, G\left(\mathbf{x}_{1:t}\right)\right]\right)\right). \tag{2.11}$$

$\mathcal{L}_{\text{GAN}}$, in addition to $\mathcal{L}_{\text{img}}$, allows our network to not only generate the target sequence, but also simultaneously enforce realism in the images through visual sharpness that fools the human eye. Note that our model uses its predictions as input for the next time-step during the training, which enables the gradients to flow through time and makes the network robust for error propagation during prediction.

2.6 Experiments

In this section, we present experiments using our network for video generation. We first evaluate our network, MCnet, on the KTH (Schuldt et al., 2004) and Weizmann action (Gorelick et al., 2007) datasets, and compare against a baseline convolutional LSTM (ConvLSTM) (Shi et al., 2015). We then proceed to evaluate on the more challenging UCF-101 (Soomro et al., 2012) dataset, in which we compare against the same ConvLSTM baseline and also the current state-of-the-art method by Mathieu et al. (2015). For all our experiments, we use $\alpha = 1$, $\lambda = 1$, and $p = 2$ in the loss functions.

In addition to the results in this section, we also provide more qualitative comparisons in the appendix and in the videos on the project website: `https://sites.google.com/a/umich.edu/rubenevillegas/iclr2017`.

Architectures. The content encoder of MCnet is built with the same architecture as VGG16 (Simonyan and Zisserman, 2015a) up to the third pooling layer. The motion encoder of MCnet is also similar to VGG16 up to the third pooling layer, except that we replace its consecutive 3x3 convolutions with single 5x5, 5x5, and 7x7 convolutions in each layer. The combination layers are composed of 3 consecutive 3x3 convolutions (256, 128, and 256 channels in each layer). The multi-scale residuals are composed of 2 consecutive 3x3 convolutions. The decoder is the mirrored architecture of the content encoder where we perform unpooling followed by deconvolution. For the baseline ConvLSTM, we use the same architecture as the motion encoder, residual connections, and decoder, except we increase the number of channels in the encoder in order to have an overall comparable number of parameters with MCnet.

2.6.1 KTH and Weizmann action datasets

Experimental settings. The KTH human action dataset (Schuldt et al., 2004) contains 6 categories of periodic motions on a simple background: running, jogging, walking, boxing,

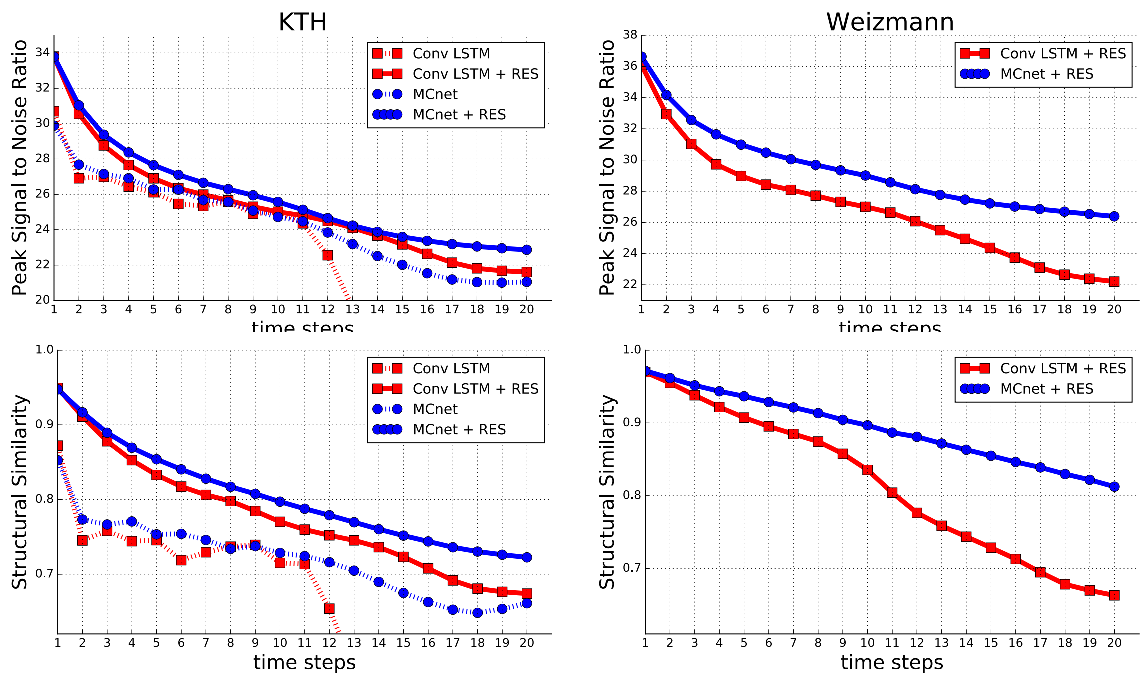


Figure 2.2: Quantitative comparison between MCnet and ConvLSTM baseline with and without multi-scale residual connections (indicated by "+ RES"). Given 10 input frames, the models predict 20 frames recursively, one by one. Left column: evaluation on KTH dataset (Schuldt et al., 2004). Right colum: evaluation on Weizmann (Gorelick et al., 2007) dataset.

hand-clapping and hand-waiving. We use person 1-16 for training and 17-25 for testing, and resize frames to 128x128 pixels. We train our network and baseline by observing 10 frames and predicting 10 frames into the future on the KTH dataset. We set $\beta = 0.02$ for training. We select the walking, running, one-hand waving, and two-hands waving sequences from the Weizmann action dataset (Gorelick et al., 2007) for testing the networks' generalizability.

For all the experiments, we test the networks on predicting 20 time steps into the future. As for evaluation, we use the same SSIM and PSNR metrics as in Mathieu et al. (2015). The evaluation on KTH was performed on sub-clips within each video in the testset. We sample sub-clips every 3 frames for running and jogging, and sample sub-clips every 20 frames (skipping the frames we have already predicted) for walking, boxing, hand-clapping, and hand-waving. Sub-clips for running, jogging, and walking were manually trimmed to ensure humans are always present in the frames. The evaluation on Weizmann was performed on all sub-clips in the selected sequences.

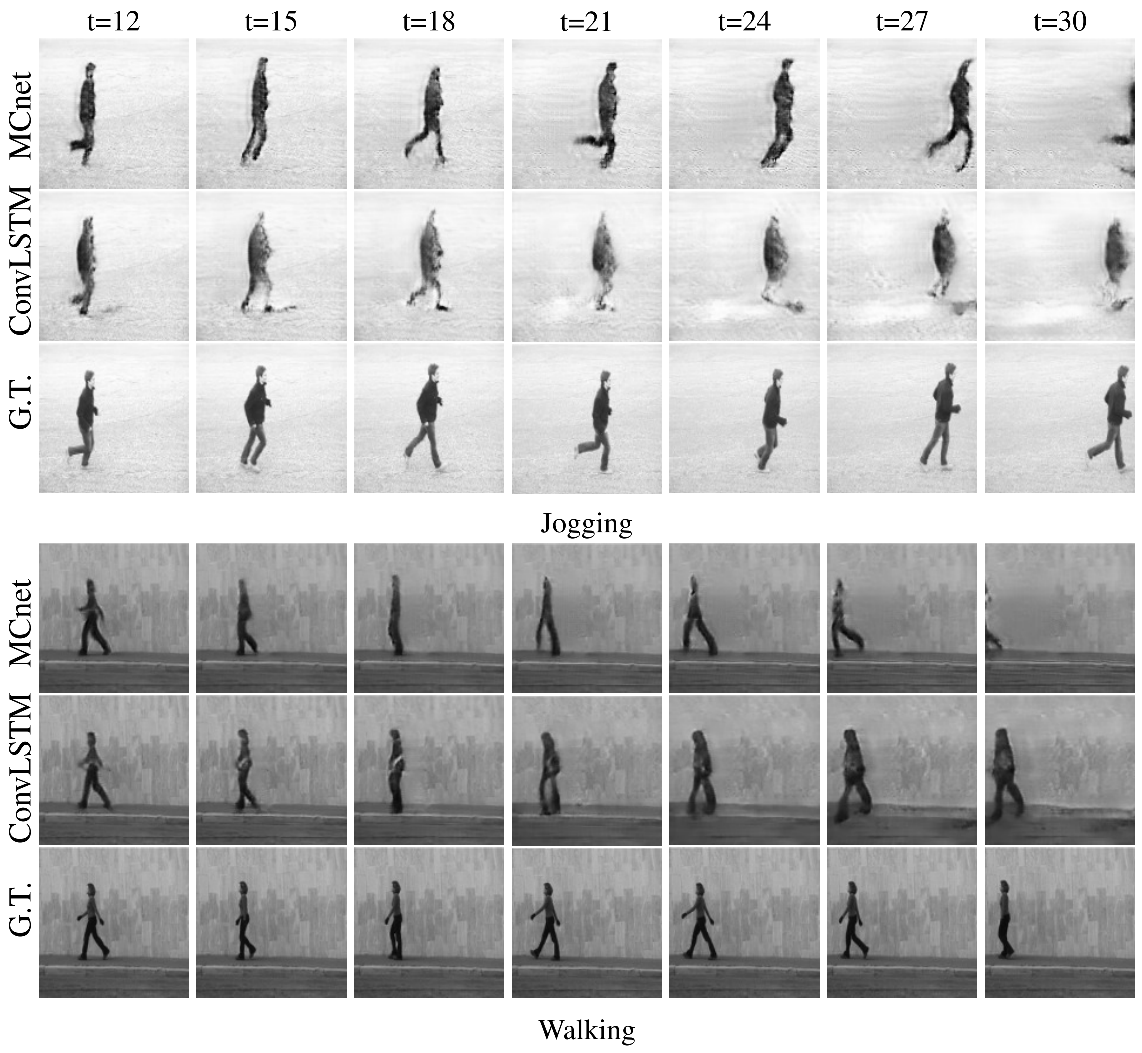


Figure 2.3: Qualitative comparison between our MCNet model and ConvLSTM. We display predictions starting from the 12th frame, in every 3 timesteps. The first 3 rows correspond to KTH dataset for the action of jogging and the last 3 rows correspond to Weizmann dataset walking action.

Results. Figure 2.2 summarizes the quantitative comparisons among our MCnet, ConvLSTM baseline and their residual variations. In the KTH test set, our network outperforms the ConvLSTM baseline by a small margin. However, when we test the residual versions of MCnet and ConvLSTM on the dataset (Gorelick et al., 2007) with similar motions, we can see that our network can generalize well to the unseen contents by showing clear improvements, especially in long-term prediction. One reason for this result is that the test and training partitions of the KTH dataset have simple and similar image contents so that ConvLSTM can memorize the average background and human appearance to make reasonable predictions. However, when tested on unseen data, ConvLSTM has to internally take care of both scene dynamics and image contents in a mingled representation, which gives it a hard time for generalization. In contrast, the reason our network outperforms the ConvLSTM baseline on unseen data is that our network focuses on identifying general motion features and applying them to a learned content representation.

Figure 2.3 presents qualitative results of multi-step prediction by our network and ConvLSTM. As expected, prediction results by our full architecture preserves human shapes more accurately than the baseline. It is worth noticing that our network produces very sharp prediction over long-term time steps; it shows that MCnet is able to capture periodic motion cycles, which reduces the uncertainty of future prediction significantly. More qualitative comparisons are shown in the appendix and the project website.

2.6.2 UCF-101 dataset

Experimental settings. This section presents results on the challenging real-world videos in the UCF-101 (Soomro et al., 2012) dataset. Having collected from YouTube, the dataset contains 101 realistic human actions taken in a wild and exhibits various challenges, such as background clutter, occlusion, and complicated motion. We employed the same network architecture as in the KTH dataset, but resized frames to 240x320 pixels, and trained the network to observe 4 frames and predict a single frame. We set $\beta = 0.001$ for training. We

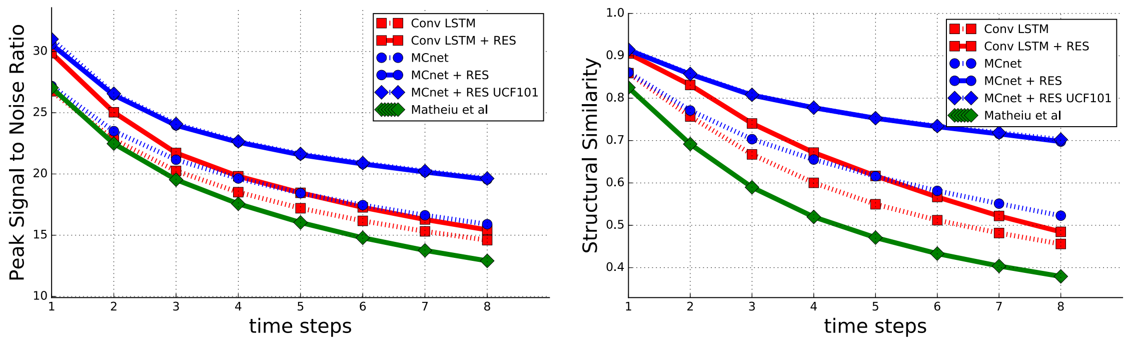


Figure 2.4: Quantitative comparison between our model, convolutional LSTM Shi et al. (2015), and Mathieu et al. (2015). Given 4 input frames, the models predict 8 frames recursively, one by one.

also trained our convolutional LSTM baseline in the same way. Following the same protocol as Mathieu et al. (2015) for data pre-processing and evaluation metrics on full images, all networks were trained on Sports-1M (Karpathy et al., 2014) dataset and tested on UCF-101 unless otherwise stated.[1]

Results. Figure 2.4 shows the quantitative comparisons between our network trained for single-step-prediction and Mathieu et al. (2015). We can clearly see the advantage of our network over the baseline. The separation of motion and contents in two encoder pathways allows our network to identify key motion and content features, which are then fed into the decoder to yield predictions of higher quality compared to the baseline.[2] In other words, our network only moves what shows motion in the past, and leaves the rest untouched.

We also trained a residual version of MCnet on UCF-101, indicated by “MCnet + RES UCF101", to compare how well our model generalizes when trained and tested on the same or different dataset(s). To our surprise, when tested with UCF-101, the MCnet trained on Sports-1M (MCnet + RES) roughly matches the performance of the MCnet trained on UCF-101 (MCnet + RES UCF101), which suggests that our model learns effective representations which can generalize to new datasets. Figure 2.5 presents qualitative comparisons between frames generated by our network and Mathieu et al. (2015). Since the ConvLSTM and

[1]We use the code and model released by Mathieu et al. (2015) at `https://github.com/coupriec/VideoPredictionICLR2016`

[2]We were not able to get the model fine-tuned on UCF-101 from the authors so it is not included in Figure 2.4

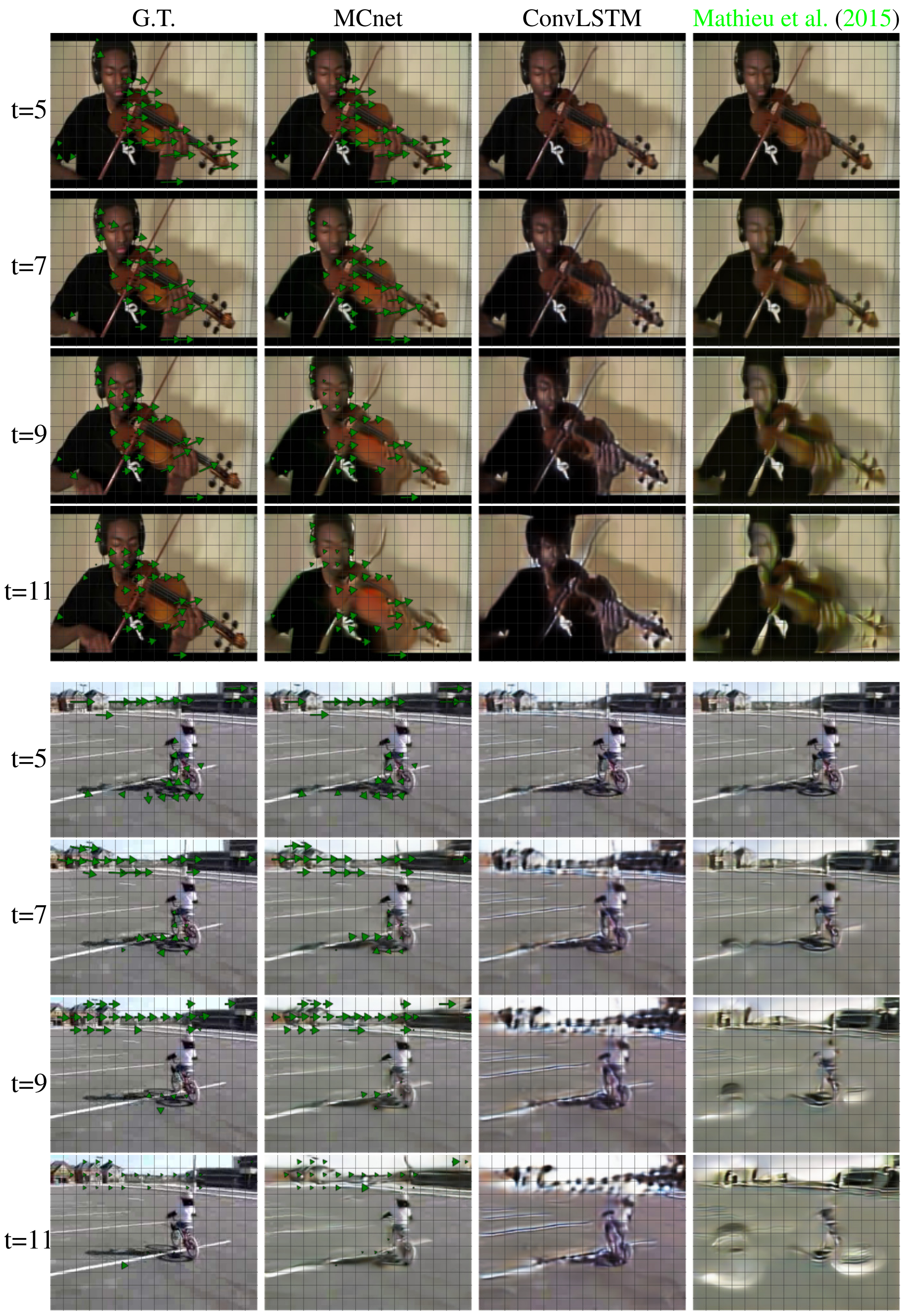


Figure 2.5: Qualitative comparisons among MCnet and ConvLSTM and Mathieu et al. (2015). We display predicted frames (in every other frame) starting from the 5^{th} frame. The green arrows denote the top-30 closest optical flow vectors within image patches between MCnet and ground-truth. More clear motion prediction can be seen in the project website.

Mathieu et al. (2015) lack explicit motion and content modules, they lose sense of the dynamics in the video and therefore the contents become distorted quickly. More qualitative comparisons are shown in the appendix and the project website.

2.7 Conclusion

We proposed a motion-content network for pixel-level prediction of future frames in natural video sequences. The proposed model employs two separate encoding pathways, and learns to decompose motion and content without explicit constraints or separate training. Experimental results suggest that separate modeling of motion and content improves the quality of the pixel-level future prediction, and our model overall achieves state-of-the-art performance in predicting future frames in challenging real-world video datasets.

CHAPTER III

Learning to Generate Long-term Future via Hierarchical Prediction

We propose a hierarchical approach for making long-term predictions of future frames. To avoid inherent compounding errors in recursive pixel-level prediction, we propose to first estimate high-level structure in the input frames, then predict how that structure evolves in the future, and finally by observing a single frame from the past and the predicted high-level structure, we construct the future frames without having to observe any of the pixel-level predictions. Long-term video prediction is difficult to perform by recurrently observing the predicted frames because the small errors in pixel space exponentially amplify as predictions are made deeper into the future. Our approach prevents pixel-level error propagation from happening by removing the need to observe the predicted frames. Our model is built with a combination of LSTM and analogy-based encoder-decoder convolutional neural networks, which independently predict the video structure and generate the future frames, respectively. In experiments, our model is evaluated on the Human 3.6M and Penn Action datasets on the task of long-term pixel-level video prediction of humans performing actions and demonstrate significantly better results than the state-of-the-art.

3.1 Introduction

Learning to predict the future has emerged as an important research problem in machine learning and artificial intelligence. Given the great progress in recognition (e.g., Krizhevsky et al. (2012); Szegedy et al. (2015)), prediction becomes an essential module for intelligent agents to plan actions or to make decisions in real-world application scenarios Jayaraman and Grauman (2015, 2016); Finn et al. (2016b). For example, robots can quickly learn manipulation skills when predicting the consequences of physical interactions. Also, an autonomous car can brake or slow down when predicting a person walking across the driving lane. In this paper, we investigate long-term future frame prediction that provides full descriptions of the visual world.

Recent recursive approaches to pixel-level video prediction highly depend on observing the generated frames in the past to make predictions further into the future Oh et al. (2015); Mathieu et al. (2016); Goroshin et al. (2015); Srivastava et al. (2015); Ranzato et al. (2014); Finn et al. (2016b); Villegas et al. (2017a); Lotter et al. (2017). In order to make reasonable long-term frame predictions in natural videos, these approaches need to be highly robust to pixel-level noise. However, the noise amplifies quickly through time until it overwhelms the signal. It is common that the first few prediction steps are of decent quality, but then the prediction degrades dramatically until all the video context is lost. Other existing works focus on predicting high-level semantics, such as trajectories or action labels Walker et al. (2014); Chao et al. (2017); Yuen and Torralba (2010b); Lee (2015), driven by immediate applications (e.g., video surveillance). We note that such high-level representations are the major factors for explaining the pixel variations into the future. In this work, we assume that the high-dimensional video data is generated from low-dimensional high-level structures, which we hypothesize will be critical for making long-term visual predictions. Our main contribution is the hierarchical approach for video prediction that involves generative modeling of video using high-level structures. Concretely, our algorithm first estimates high-level structures of observed frames, and then predicts their future states, and finally generates future frames conditioned on predicted high-level structures.

The prediction of future structure is performed by an LSTM that observes a sequence of structures estimated by a CNN, encodes the observed dynamics, and predicts the future sequence of such structures. We note that Fragkiadaki et al. (2015a) developed an LSTM architecture that can straightforwardly be adapted to our method. However, our main contribution is the hierarchical approach for video prediction, so we choose a simpler LSTM architecture to convey our idea. Our approach then observes a single frame from the past and predicts the entire future described by the predicted structure sequence using an analogy-making network Reed et al. (2015a). In particular, we propose an image generator that learns a shared embedding between image and high-level structure information which allows us convert an input image into a future image guided by the structure difference between the input image and the future image. We evaluate the proposed model on challenging real-world human action video datasets. We use 2D human poses as our high-level structures similar to Reed et al. (2016a). Thus, our LSTM network models the dynamics of human poses while our analogy-based image generator network learns a joint image-pose embedding that allows the pose difference between an observed frame and a predicted frame to be transferred to image domain for future frame generation. As a result, this pose-conditioned generation strategy prevents our network from propagating prediction errors through time, which in turn leads to very high quality future frame generation for long periods of time. Overall, the promising results of our approach suggest that it can be greatly beneficial to incorporate proper high-level structures into the generative process.

The rest of the paper is organized as follows: A review of the related work is presented in Section 3.2. The overview of the proposed algorithm is presented in Section 3.3. The network configurations and their training algorithms are described in Section 3.4 and Section 3.5, respectively. We present the experimental details and results in Section 3.6, and conclude the paper with discussions of future work in Section 3.7.

3.2 Related Work

Early work on future frame prediction focused on small patches containing simple predictable motions Sutskever et al. (2009); Michalski et al. (2014a); Mittelman et al. (2014) and motions in real videos Ranzato et al. (2014); Srivastava et al. (2015). High resolution videos contain far more complicated motion which cannot be modeled in a patch-wise manner due to the well known aperture problem. The aperture problem causes blockiness in predictions as we move forward in time. Ranzato et al. (2014) tried to solve blockiness by averaging over spatial displacements after predicting patches; however, this approach does not work for long-term predictions.

Recent approaches in video prediction have moved from predicting patches to full frame prediction. Oh et al. (2015) proposed a network architecture for action conditioned video prediction in Atari games. Mathieu et al. (2016) proposed an adversarial loss for video prediction and a multi-scale network architecture that results in high quality prediction for a few timesteps in natural video; however, the frame prediction quality degrades quickly. Finn et al. (2016b) proposed a network architecture to directly transform pixels from a current frame into the next frame by predicting a distribution over pixel motion from previous frames. Xue et al. (2016) proposed a probabilistic model for predicting possible motions of a single input frame by training a motion encoder in a variational autoencoder approach. Vondrick et al. (2016) built a model that generates realistic looking video by separating background and foreground motion. Villegas et al. (2017a) improved the convolutional encoder/decoder architecture by separating motion and content features. Lotter et al. (2017) built an architecture inspired by the predictive coding concept in neuroscience literature that predicts realistic looking frames.

All the previously mentioned approaches attempt to perform video generation in a pixel-to-pixel process. We aim to perform the prediction of future frames in video by taking a hierarchical approach of first predicting the high-level structure and then using the predicted

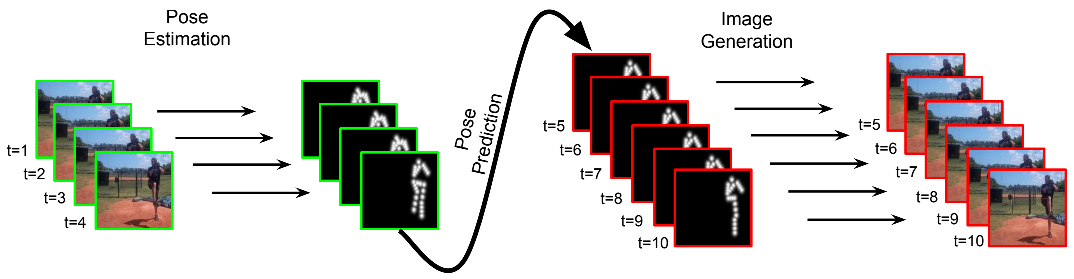


Figure 3.1: Overall hierarchical approach to pixel-level video prediction. Our algorithm first observes frames from the past and estimate the high-level structure, in this case human pose xy-coordinates, in each frame. The estimated structure is then used to predict the future structures in a sequence to sequence manner. Finally, our algorithm takes the last observed frame, its estimated structure, and the predicted structure sequence, in this case represented as heatmaps, and generates the future frames. Green denotes input to our network and red denotes output from our network.

structure to predict the future in the video from a single frame input.

To the best of our knowledge, this is the first hierarchical approach to pixel-level video prediction. Our hierarchical architecture makes it possible to generate good quality long-term predictions that outperform current approaches. The main success from our algorithm comes from the novel idea of first making high-level structure predictions which allows us to observe a single image and generate the future video by visual-structure analogy. Our image generator learns a shared embedding between image and structure inputs that allows us to transform high-level image features into a future image driven by the predicted structure.

3.3 Overview

This paper tackles the task of long-term video prediction in a hierarchical perspective. Given the input high-level structure sequence $\mathbf{p}_{1:t}$ and frame $\mathbf{x}_t$, our algorithm is asked to predict the future structure sequence $\mathbf{p}_{t+1:t+T}$ and subsequently generate frames $\mathbf{x}_{t+1:t+T}$. The problem with video frame prediction originates from modeling pixels directly in a sequence-to-sequence manner and attempting to generate frames in a recurrent fashion. Current state-of-the-art approaches recurrently observe the predicted frames, which causes rapidly increasing error accumulation through time. Our objective is to avoid having to observe generated future frames at all during the full video prediction procedure.

Figure 3.1 illustrates our hierarchical approach. Our full pipeline consists of 1) performing high-level structure estimation from the input sequence, 2) predicting a sequence of future high-level structures, and 3) generating future images from the predicted structures by visual-structure analogy-making given an observed image and the predicted structures. We explore our idea by performing pixel-level video prediction of human actions while treating human pose as the high-level structure. Hourglass network Newell et al. (2016) is used for pose estimation on input images. Subsequently, a sequence-to-sequence LSTM-recurrent network is trained to read the outputs of hourglass network and to predict the future pose sequence. Finally, we generate the future frames by analogy making using the pose relationship in feature space to transform the last observed frame.

The proposed algorithm makes it possible to decompose the task of video frame prediction to sub-tasks of future high-level structure prediction and structure-conditioned frame generation. Therefore, we remove the recursive dependency of generated frames that causes the compound errors of pixel-level prediction in previous methods, and so our method performs very long-term video prediction.

3.4 Architecture

This section describes the architecture of the proposed algorithm using human pose as a high-level structure. Our full network is composed of two modules: an encoder-decoder LSTM that observes and outputs xy-coordinates, and an image generator that performs visual analogy based on high-level structure heatmaps constructed from the xy-coordinates output from LSTM.

3.4.1 Future Prediction of High-Level Structures

Figure 3.2 illustrates our pose predictor. Our network first encodes the observed structure dynamics by

$$[\mathbf{h}_t, \mathbf{c}_t] = \text{LSTM}\left(\mathbf{p}_t, \mathbf{h}_{t-1}, \mathbf{c}_{t-1}\right), \tag{3.1}$$

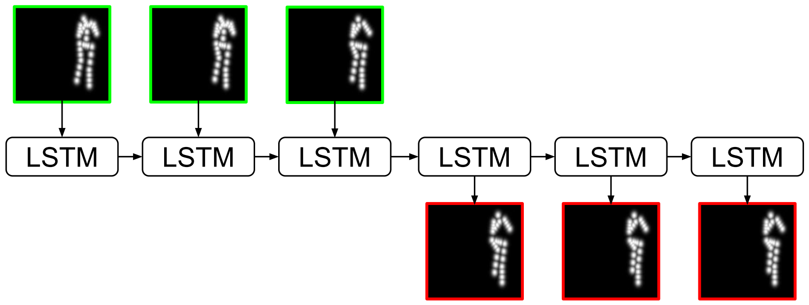


Figure 3.2: Illustration of our pose predictor. LSTM observes k consecutive human pose inputs and predicts the pose for the next T timesteps. Note that the human heatmaps are used for illustration purposes, but our network observes and outputs xy-coordinates.

where $\mathbf{h}_t \in \mathbb{R}^H$ represents the observed dynamics up to time t, $\mathbf{c}_t \in \mathbb{R}^H$ is the *memory cell* that retains information from the history of pose inputs, $\mathbf{p}_t \in \mathbb{R}^{2L}$ is the pose at time t (i.e., 2D coordinate positions of L joints). In order to make a reasonable prediction of the future pose, LSTM has to first observe a few pose inputs to identify the type of motion occurring in the pose sequence and how it is changing over time. LSTM also has to be able to remove noise present in the input pose, which can come from annotation error if using the dataset-provided pose annotation or pose estimation error if using a pose estimation algorithm. After a few pose inputs have been observed, LSTM generates the future pose by

$$\hat{\mathbf{p}}_t = f\left(\mathbf{w}^\top \mathbf{h}_t\right), \tag{3.2}$$

where $\mathbf{w}$ is a projection matrix, f is a function on the projection (i.e. tanh or identity), and $\hat{\mathbf{p}}_t \in \mathbb{R}^{2L}$ is the predicted pose. In the subsequent predictions, our LSTM does not observe the previously generated pose. Not observing generated pose in LSTM prevents errors in the pose prediction from being propagated into the future, and it also encourages the LSTM internal representation to contain robust high-level features that allow it to generate the future sequence from only the original observation. As a result, the representation obtained in the pose input encoding phase must obtain all the necessary information for generating the correct action sequence in the decoding phase. After we have set the human pose sequence

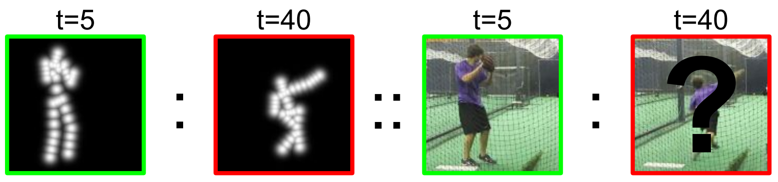


Figure 3.3: Generating image frames by making analogies between high-level structures and image pixels.

for the future frames, we proceed to generate the pixel-level visual future.

3.4.2 Image Generation by Visual-Structure Analogy

To synthesize a future frame given its pose structure, we make a visual-structure analogy inspired by Reed et al. (2015a) following $\mathbf{p}_t : \mathbf{p}_{t+n} :: \mathbf{x}_t : \mathbf{x}_{t+n}$, read as "$\mathbf{p}_t$ is to $\mathbf{p}_{t+n}$ as $\mathbf{x}_t$ is to $\mathbf{x}_{t+n}$" as illustrated in Figure 3.3. Intuitively, the future frame $\mathbf{x}_{t+n}$ can be generated by transferring the structure transformation from $\mathbf{p}_t$ to $\mathbf{p}_{t+n}$ to the observed frame $\mathbf{x}_t$. Our image generator instantiates this idea using a pose encoder f_{pose}, an image encoder f_{img} and an image decoder f_{dec}. Specifically, f_{pose} is a convolutional encoder that specializes on identifying key pose features from the pose input that reflects high-level human structure.[1] f_{img} is also a convolutional encoder that acts on an image input by mapping the observed appearance into a feature space where the pose feature transformations can be easily imposed to synthesize the future frame using the convolutional decoder f_{dec}. The visual-structure analogy is then performed by

$$\hat{\mathbf{x}}_{t+n} = f_{\text{dec}}\left(f_{\text{pose}}\left(g\left(\hat{\mathbf{p}}_{t+n}\right)\right) - f_{\text{pose}}\left(g\left(\mathbf{p}_t\right)\right) + f_{\text{img}}\left(\mathbf{x}_t\right)\right), \tag{3.3}$$

where $\hat{\mathbf{x}}_{t+n}$ and $\hat{\mathbf{p}}_{t+n}$ are the generated image and corresponding predicted pose at time $t+n$, $\mathbf{x}_t$ and $\mathbf{p}_t$ are the input image and corresponding estimated pose at time t, and $g\left(.\right)$ is a function that maps the output xy-coordinates from LSTM into L depth-concatenated

[1]Each input pose to our image generator is converted to concatenated heatmaps of each landmark before computing features.

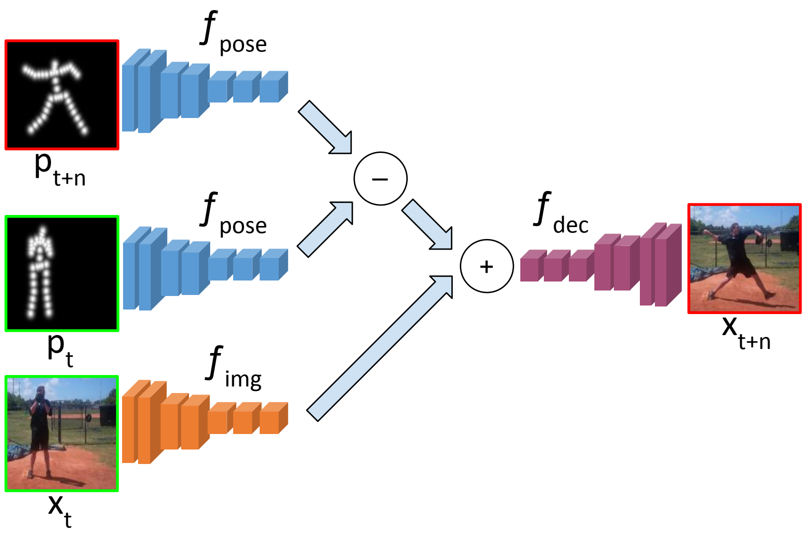


Figure 3.4: Illustration of our image generator. Our image generator observes an input image, its corresponding human pose, and the human pose of the future image. Through analogy making, our network generates the next frame.

heatmaps.[2] Intuitively, f_{pose} infers features whose "substractive" relationship is the same subtractive relationship between $\mathbf{x}_{t+n}$ and $\mathbf{x}_t$ in the feature space computed by f_{img}, i.e., $f_{\text{pose}}(g(\hat{\mathbf{p}}_{t+n})) - f_{\text{pose}}(g(\hat{\mathbf{p}}_t)) \approx f_{\text{img}}(\mathbf{x}_{t+n}) - f_{\text{img}}(\mathbf{x}_t)$.

The network diagram is illustrated in in Figure 3.4. The relationship discovered by our network allows for highly non-linear transformations between images to be inferred by a simple addition/subtraction in feature space.

3.5 Training

In this section, we first summarize the multi-step video prediction algorithm using our networks and then describe the training strategies of the high-level structure LSTM and of the visual-structure analogy network. We train our high-level structure LSTM independent from the visual-structure analogy network, but both are combined during test time to perform video prediction.

[2]We independently construct the heatmap with a Gaussian function around the xy-coordinates of each landmark.

Algorithm 1 Video Prediction Procedure

```
input: $\mathbf{x}_{1:k}$
output: $\hat{\mathbf{x}}_{k+1:k+T}$
for $t$=1 to $k$ do
    $\mathbf{p}_t \leftarrow \text{Hourglass}(\mathbf{x}_t)$
    $[\mathbf{h}_t, \mathbf{c}_t] \leftarrow \text{LSTM}(\mathbf{p}_t, \mathbf{h}_{t-1}, \mathbf{c}_{t-1})$
end for
for $t$=$k+1$ to $k+T$ do
    $[\mathbf{h}_t, \mathbf{c}_t] \leftarrow \text{LSTM}(\mathbf{h}_{t-1}, \mathbf{c}_{t-1})$
    $\hat{\mathbf{p}}_t \leftarrow f\left(\mathbf{w}^\top \mathbf{h}_t\right)$
    $\hat{\mathbf{x}}_t \leftarrow f_{\text{dec}}\left(f_{\text{pose}}\left(g\left(\hat{\mathbf{p}}_t\right)\right) - f_{\text{pose}}\left(g\left(\mathbf{p}_k\right)\right) + f_{\text{img}}\left(\mathbf{x}_k\right)\right)$
end for
```

3.5.1 Multi-Step Prediction

Our algorithm multi-step video prediction procedure is described in Algorithm 1. Given input video frames, we use the Hourglass network Newell et al. (2016) to estimate the human poses $\mathbf{p}_{1:k}$. High-level structure LSTM then observes $\mathbf{p}_{1:k}$, and proceeds to generate a pose sequence $\hat{\mathbf{p}}_{k+1:k+T}$ where T is the desired number of time steps to predict. Next, our visual-structure analogy network takes $\mathbf{x}_k$, $\mathbf{p}_k$, and $\hat{\mathbf{p}}_{k+1:k+T}$ and proceeds to generate future frames $\hat{\mathbf{x}}_{k+1:k+T}$ one by one. Note that the future frame prediction is performed by observing pixel information from only $\mathbf{x}_k$, that is, we never observe any of the predicted frames.

3.5.2 High-Level Structure LSTM Training

We employ a sequence-to-sequence approach to predict the future structures (i.e. future human pose). Our LSTM is *unrolled* for k timesteps to allow it to observe k pose inputs before making any prediction. Then we minimize the prediction loss defined by

$$\mathcal{L}_{\text{pose}} = \frac{1}{TL} \sum_{t=1}^{T} \sum_{l=1}^{L} \mathbb{1}_{\{m^l_{k+t}=1\}} \|\hat{\mathbf{p}}^l_{k+t} - \mathbf{p}^l_{k+t}\|_2^2, \tag{3.4}$$

where $\hat{\mathbf{p}}^l_{k+t}$ and $\mathbf{p}^l_{k+t}$ are the predicted and ground-truth pose l-th landmark, respectively, $\mathbb{1}_{\{.\}}$ is the indicator function, and m^l_{k+t} tells us whether a landmark is visible or not (i.e. not present in the ground-truth). Intuitively, the indicator function allows our LSTM to make a guess of the non-visible landmarks even when not present at training. Even in the absence of a few landmarks during training, LSTM is able to internally understand the human structure and observed motion. Our training strategy allows LSTM to make a reasonable guess of the landmarks not present in the training data by using the landmarks available as context.

3.5.3 Visual-Structure Analogy Training

Training our network to transform an input image into a target image that is too close in image space can lead to suboptimal parameters being learned due to the simplicity of such task that requires only changing a few pixels. Because of this, we train our network to perform random *jumps in time* within a video clip. Specifically, we let our network observe a frame $\mathbf{x}_t$ and its corresponding human pose $\mathbf{p}_t$, and force it to generate frame $\mathbf{x}_{t+n}$ given pose $\mathbf{p}_{t+n}$, where n is defined randomly for every iteration at training time. Training to jump to random frames in time gives our network a clear signal the task at hand due to the large pixel difference between frames far apart in time.

To train our network, we use the compound loss from Dosovitskiy and Brox (2016a). Our network is optimized to minimize the objective given by

$$\mathcal{L} = \mathcal{L}_{\text{img}} + \mathcal{L}_{\text{feat}} + \mathcal{L}_{\text{Gen}}, \tag{3.5}$$

where $\mathcal{L}_{\text{img}}$ is the loss in image space defined by

$$\mathcal{L}_{\text{img}} = \|\mathbf{x}_{t+n} - \hat{\mathbf{x}}_{t+n}\|_2^2, \tag{3.6}$$

where $\mathbf{x}_{t+n}$ and $\hat{\mathbf{x}}_{t+n}$ are the target and predicted frames, respectively. The image loss intuitively guides our network towards a rough blurry pixel-leven frame prediction that reflects most details of the target image. $\mathcal{L}_{\text{feat}}$ is the loss in feature space define by

$$\begin{aligned}\mathcal{L}_{\text{feat}} &= \left\|C_1\left(\mathbf{x}_{t+n}\right) - C_1\left(\hat{\mathbf{x}}_{t+n}\right)\right\|_2^2 \\ &+ \left\|C_2\left(\mathbf{x}_{t+n}\right) - C_2\left(\hat{\mathbf{x}}_{t+n}\right)\right\|_2^2,\end{aligned} \tag{3.7}$$

where $C_1(.)$ extracts features representing mostly image appearance, and $C_2(.)$ extracts features representing mostly image structure. Combining appearance sensitive features with structure sensitive features gives our network a learning signal that allows it to make frame predictions with accurate appearance while also enforcing correct structure. $\mathcal{L}_{\text{Gen}}$ is the term in adversarial loss that allows our model to generate realistic looking images and is defined by

$$\mathcal{L}_{\text{Gen}} = -\log D\left(\left[\mathbf{p}_{t+n}, \hat{\mathbf{x}}_{t+n}\right]\right), \tag{3.8}$$

where $\hat{\mathbf{x}}_{t+n}$ is the predicted image, $\mathbf{p}_{t+n}$ is the human pose corresponding to the target image, and $D(.)$ is the discriminator network in adversarial loss. This sub-loss allows our network to generate images that reflect a similar level of detail as the images observed in the training data.

During the optimization of D, we use the mismatch term proposed by Reed et al. (2016b), which allows the discriminator D to become sensitive to mismatch between the generation and the condition. The discriminator loss is defined by

$$\begin{aligned}\mathcal{L}_{\text{Disc}} &= -\log D\left(\left[\mathbf{p}_{t+n}, \mathbf{x}_{t+n}\right]\right) \\ &- 0.5\log\left(1 - D\left(\left[\mathbf{p}_{t+n}, \hat{\mathbf{x}}_{t+n}\right]\right)\right) \\ &- 0.5\log\left(1 - D\left(\left[\mathbf{p}_{t+n}, \mathbf{x}_{t}\right]\right)\right),\end{aligned} \tag{3.9}$$

while optimizing our generator with respect to the adversarial loss, the mismatch-aware term sends a stronger signal to our generator resulting in higher quality image generation, and network optimization. Essentially, having a discriminator that knows the correct structure-image relationship, reduces the parameter search space of our generator while optimizing to fool the discriminator into believing the generated image is real. The latter in combination with the other loss terms allows our network to produce high quality image generation given the structure condition.

3.6 Experiments

In this section, we present experiments on pixel-level video prediction of human actions on the Penn Action Zhang et al. (2013) and Human 3.6M datasets Ionescu et al. (2014a). Pose landmarks and video frames are normalized to be between -1 and 1, and frames are cropped based on temporal tubes to remove as much background as possible while making sure the human of interest is in all frames. For the feature similarity loss term (Equation 3.7), we use we use the last convolutional layer in AlexNet Krizhevsky et al. (2012) as C_1, and the last layer of the Hourglass Network in Newell et al. (2016) as C_2. We augmented the available video data by performing horizontal flips randomly at training time for Penn Action. Motion-based pixel-level quantitative evaluation using Peak Signal-to-Noise Ratio (PSNR), analysis, and control experiments can be found in the appendix. For video illustration of our method, please refer to the project website: https://sites.google.com/a/umich.edu/rubenevillegas/hierch_vid.

We compare our method against two baselines based on convolutional LSTM and optical flow. The convolutional LSTM baseline Shi et al. (2015) was trained with adversarial loss Mathieu et al. (2016) and the feature similarity loss (Equation 3.7). An optical flow based baseline used the last observed optical flow Farnebäck (2003) to move the pixels of the last observed frame into the future.

We follow a human psycho-physical quantitative evaluation metric similar to Vondrick et al. (2016). Amazon Mechanical Turk (AMT) workers are given a two-alternative choice to indicate which of two videos looks more realistic. Specifically, the workers are shown a pair of videos (generated by two different methods) consisting of the same input frames indicated by a green box and predicted frames indicated by a red box, in addition to the action label of the video. The workers are instructed to make their decision based on the frames in the red box. Additionally, we train a Two-stream action recognition network Simonyan and Zisserman (2014) on the Penn Action dataset and test on the generated videos to evaluate if our network is able to generate videos predicting the activities observed in the original dataset. We do not perform action classification experiments on the Human 3.6M dataset due to high uncertainty in the human movements and high motion similarity amongst actions.

Architectures. The sequence prediction LSTM is made of a single layer encoder-decoder LSTM with tied parameters, 1024 hidden units, and tanh output activation. Note that the decoder LSTM does not observe any inputs other than the hidden units from the encoder LSTM as initial hidden units. The image and pose encoders are built with the same architecture as VGG16 Simonyan and Zisserman (2015a) up to the third pooling layer, except that the pose encoder takes in the pose heat-maps as an image made of L channels, and the image encoder takes a regular 3-channel image. The decoder is the mirrored architecture of the image encoder where we perform unpooling followed by deconvolution, and a final tanh activation. The convolutional LSTM baseline is built with the same architecture as the image encoder and decoder, but there is a convolutional LSTM layer with the same kernel size and number of channels as the last layer in the image encoder connecting them.

3.6.1 Penn Action Dataset

Experimental setting. The Penn Action dataset is composed of 2326 video sequences of 15 different actions and 13 human joint annotations for each sequence. To train our image generator, we use the standard train split provided in the dataset. To train our pose predictor, we sub-sample the actions in the standard train-test split due to very noisy joint ground-truth. We used videos from the actions of baseball pitch, baseball swing, clean and jerk, golf swing, jumping jacks, jump rope, tennis forehand, and tennis serve. Our pose predictor is trained to observe 10 inputs and predict 32 steps, and tested on predicting up to 64 steps (some videos'

"Which video is more realistic?"	B.B.	C. & J.	G.	J. J.	J. R.	T.	Mean
Prefers ours over Convolutional LSTM	89.5%	87.2%	84.7%	83.0%	66.7%	88.2%	82.4%
Prefers ours over Optical Flow	87.8%	86.5%	80.3%	88.9%	86.2%	85.6%	86.1%

Table 3.1: Penn Action Video Generation Preference: We show videos from two methods to Amazon Mechanical Turk workers and ask them to indicate which is more realistic. The table shows the percentage of times workers preferred our model against baselines. A majority of the time workers prefer predictions from our model. We merged baseball pitch and baseball swing into baseball, and tennis forehand and tennis serve into tennis.

groundtruth end before 64 steps). Our image generator is trained to make single random jumps within 30 steps into the future. Our evaluations are performed on a single clip that starts at the first frame of each video.

AMT results. These experiments were performed by 66 unique workers, where a total of 1848 comparisons were made (934 against convolutional LSTM and 914 against optical flow baseline). As shown in Table 3.1 and Figure 3.5, our method is capable of generating more realistic sequences compared to the baselines. Quantitatively, the action sequences generated by our network are perceptually higher quality than the baselines and also predict the correct action sequence. A relatively small (although still substantial) margin is observed when comparing to convolutional LSTM for the jump rope action (i.e., 66.7% for ours vs 33.3% for Convolutional LSTM). We hypothesize that convolutional LSTM is able to do a reasonable job for this action class due the highly cyclic motion nature of jumping up and down in place. The remainder of the human actions contain more complicated non-linear motion, which is much more complicated to predict. Overall, our method outperforms the baselines by a large margin (i.e. 82.4% for ours vs 17.6% for Convolutional LSTM, and 86.1% for ours vs 13.9% for Optical Flow). Side by side video comparison for all actions can be found in our project website.

Action recognition results. To see whether the generated videos contain actions that can fool a CNN trained for action recognition, we train a Two-Stream CNN on the PennAction dataset. In Table 3.2, "Temporal Stream" denotes the network that observes motion as

Method	Temporal Stream	Spatial Stream	Combined
Real Test Data *	66.6%	63.3%	72.1%
Ours	**35.7**%	**52.7**%	**59.0**%
Convolutional LSTM	13.9%	45.1%	46.4%
Optical Flow	13.9%	39.2%	34.9%

Table 3.2: Activity recognition evaluation.

concatenated optical flow (Farneback's optical flow) images as input, and "Spatial Stream" denotes the network that observes single image as input. "Combined" denotes the averaging of the output probability vectors from the Temporal and Spatial stream. "Real test data" denotes evaluation on the ground-truth videos (i.e. perfect prediction).

From Table 3.2, it is shown that our network is able to generate videos that are far more representative of the correct action compared to all baselines, in both Temporal and Spatial stream, regardless of using a neural network as the judge. When combining both Temporal and Spatial streams, our network achieves the best quality videos in terms of making a pixel-level prediction of the correct action.

Pixel-level evaluation and control experiments. We evaluate the frames generated by our method using PSNR as measure, and separated the test data based on amount of motion, as suggested by Villegas et al. (2017a). From these experiments, we conclude that pixel-level evaluation highly depends on predicting the exact future observed in the ground-truth. Highest PSNR scores are achieved when trajectories of the exact future is used to generate the future frames. Due to space constraints, we ask the reader to please refer to the appendix for more detailed quantitative and qualitative analysis.

3.6.2 Human 3.6M Dataset

Experimental settings. The Human 3.6M dataset Ionescu et al. (2014a) is composed of 3.6 million 3D human poses (we use the provided 2D pose projections) composed of 32 joints and corresponding images taken from 11 professional actors in 17 scenarios. For training, we use subjects number 1, 5, 6, 7, and 8, and test on subjects number 9 and 11. Our pose predictor is trained to observe 10 inputs and predict 64 steps, and tested on predicting 128 steps. Our image generator is trained to make single random jumps anywhere in the training videos. We evaluate on a single clip from each test video that starts at the exact middle of the video to make sure there is motion occurring.

AMT results. We collected a total of 2203 comparisons (1086 against convolutional LSTM and 1117 against optical flow baseline) from 71 unique workers. As shown in Table 3.3, the videos generated by our network are perceptually higher quality and reflect a reasonable future compared to the baselines on average. Unexpectedly, our network does not perform well on videos where the action involves minimal motion, such as sitting, sitting down, eating, taking a photo, and waiting. These actions usually involve the person staying still or making very unnoticeable motion which can result in a static prediction (by convolutional LSTM and/or optical flow) making frames look far more realistic than the prediction from our network. Overall, our method outperforms the baselines by a large margin (i.e. 70.3% for ours vs 29.7% for Convolutional LSTM, and 72.3% for ours vs 27.7% for Optical Flow). Figure 3.5 shows that our network generates far higher quality future frames compared to the convolutional LSTM baseline. Side by side video comparison for all actions can be found in our project website.

"Which video is more realistic?"	Direct.	Disc.	Eating	Greet.	Phoning	Photo	Posing
Prefers ours over Convolutional LSTM	67.6%	75.9%	74.7%	79.5%	69.7%	66.2%	69.7%
Prefers ours over Optical Flow	61.4%	89.3%	43.8%	80.3%	84.5%	52.0%	75.3%

"Which video is more realistic?"	Purch.	Sit	Sit Down	Smoke	Wait	Walk	Mean
Prefers ours over Convolutional LSTM	79.0%	38.0%	54.7%	70.4%	50.0%	86.0%	70.3%
Prefers ours over Optical Flow	85.7%	35.1%	46.7%	73.3%	84.3%	90.8%	72.3%

Table 3.3: Human 3.6M Video Generation Preference: We show videos from two methods to Amazon Mechanical Turk workers and ask them to indicate which of the the two looks more realistic. The table shows the percentage of times workers preferred our model against baselines. Most of the time workers prefer predictions from our model. We merge the activity categories of walking, walking dog, and walking together into walking.

Pixel-level evaluation and control experiments. Following the same procedure as Section 3.6.1, we evaluated the predicted videos using PSNR and separated the test data by motion. Due to the high uncertainty and number of prediction steps in these videos, the predicted future can largely deviate from the exact future observed in the ground-truth. The highest PSNR scores are again achieved when the exact future pose is used to generate the video frames; however, there is an even larger gap compared to the results in Section 3.6.1. Due to space constraints, we ask the reader to please refer to the appendix for more detailed quantitative and qualitative analysis.

3.7 Conclusion and Future Work

We propose a hierarchical approach of pixel-level video prediction. Using human action videos as benchmark, we have demonstrated that our hierarchical prediction approach is able to predict up to 128 future frames, which is an order of magnitude improvement in terms of effective temporal scale of the prediction.

The success of our approach demonstrates that it can be greatly beneficial to incorporate the proper high-level structure into the generative process. At the same time, an important open research question would be how to automatically learn such structures without domain knowledge. We leave this as future work.

Another limitation of this work is that it generates a single future trajectory. For an agent to make a better estimation of what the future looks like, we would need more than one

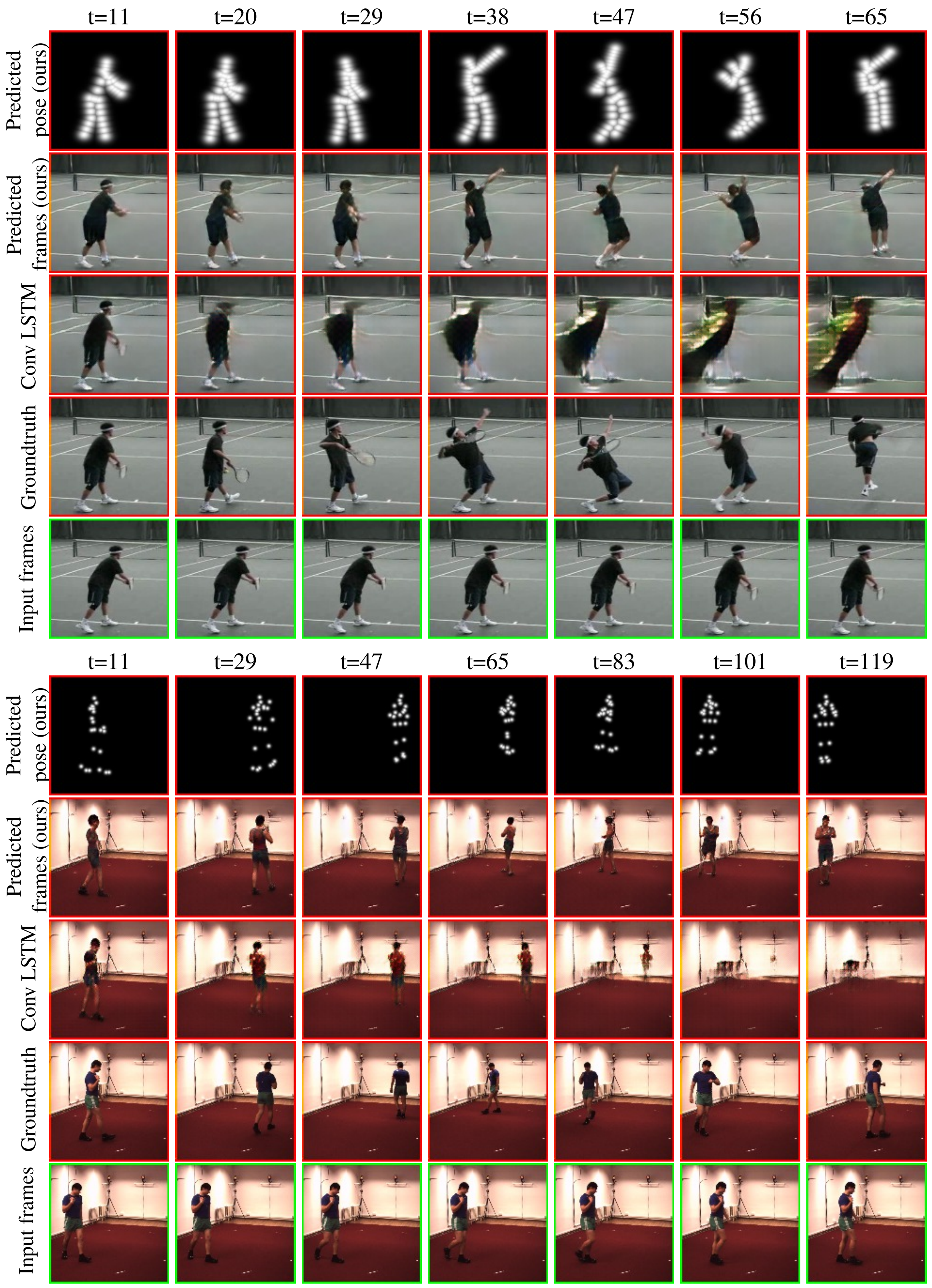


Figure 3.5: Qualitative evaluation of our network for 55 step prediction on Penn Action (top rows), and 109 step prediction on Human 3.6M (bottom rows). Our algorithm observes 10 previous input frames, estimates the human pose, predicts the pose sequence of the future, and it finally generates the future frames. Green box denotes input and red box denotes prediction. We show the last 7 input frames. Side by side video comparisons can be found in our project website.

generated future. Future work will involve the generation of many futures given using a probabilistic sequence model.

Finally, our model does not handle background motion. This is a highly challenging task since background comes in and out of sight. Predicting background motion will require a generative model that hallucinates the unseen background. We also leave this as future work.

CHAPTER IV

Hierarchical Long-term Video Prediction without Supervision

Much of recent research has been devoted to video prediction and generation, yet most of the previous works have demonstrated only limited success in generating videos on short-term horizons. The hierarchical video prediction method by Villegas et al. (2017b) is an example of a state-of-the-art method for long-term video prediction, but their method is limited because it requires ground truth annotation of high-level structures (e.g., human joint landmarks) at training time. Our network encodes the input frame, predicts a high-level encoding into the future, and then a decoder with access to the first frame produces the predicted image from the predicted encoding. The decoder also produces a mask that outlines the predicted foreground object (e.g., person) as a by-product. Unlike Villegas et al. (2017b), we develop a novel training method that jointly trains the encoder, the predictor, and the decoder together without high-level supervision; we further improve upon this by using an adversarial loss in the feature space to train the predictor. Our method can predict about 20 seconds into the future and provides better results compared to Denton and Fergus (2018) and Finn et al. (2016a) on the Human 3.6M dataset.

4.1 Introduction

Building a model that is able to predict the future states of an environment from raw high-dimensional sensory data (e.g., video) has recently emerged as an important research problem in machine learning and computer vision. Models that are able to accurately predict the future can play a vital role in developing intelligent agents that interact with their environment Jayaraman and Grauman (2015, 2016); Finn et al. (2016a).

Popular video prediction approaches focus on recursively observing the generated frames to make predictions farther into the future Oh et al. (2015); Mathieu et al. (2016); Goroshin et al. (2015); Srivastava et al. (2015); Ranzato et al. (2014); Finn et al. (2016a); Villegas et al. (2017b); Lotter et al. (2017). In order to make reasonable long-term frame predictions in natural videos, these approaches need to automatically identify the dynamics of the main factors of variation changing through time, while also being highly robust to pixel-level noise. However, it is common for the previously mentioned methods to generate quality predictions for the first few steps, but then the prediction dramatically degrades until all of the video context is lost or the predicted motion becomes static.

A hierarchical method makes predictions in a high-level information hierarchy (e.g., landmarks) and then decodes the predicted future in high-level back into low-level pixel space. The advantage of predicting the future in high-level space first is that the predictions degrade less quickly compared to predictions made solely in pixel space. The method by Villegas et al. (2017b) is an example of a hierarchical model; however, it requires ground truth human landmark annotations during training time. In this work, we explore ways to generate videos using a hierarchical model *without* requiring ground truth landmarks or other high-level structure annotations during training. In a similar fashion to Villegas et al. (2017b), the proposed network predicts the pixels of future video frames given the first few frames. Specifically, our network never observes any of the predicted frames, and the predicted future frames are driven solely by the high-level space predictions.

The contributions of our work are summarized below:

- An unsupervised approach for discovering high-level features necessary for long-term future prediction.
- A joint training strategy for generating high-level features from low-level features and low-level features from high-level features simultaneously.
- Use of adversarial training in feature space for improved high-level feature discovery and generation.
- Long-term pixel-level video prediction for about 20 seconds into the future for the Human 3.6M dataset.

4.2 Related Work

Patch-level prediction The video prediction problem was initially studied at the patch level containing synthetic motions (Sutskever et al., 2009; Michalski et al., 2014a; Mittelman et al., 2014). Srivastava et al. (2015) and Ranzato et al. (2014) followed up by proposing methods that can handle prediction in natural videos. However, predicting patches encounters the well-known aperture problem that causes blockiness as prediction advances in time.

Frame-level prediction on realistic videos. More recently, the video prediction problem has been formulated at the full frame level using convolutional encoder/decoder networks as the main component. Finn et al. (2016a) proposed a network that can perform next frame video prediction by explicitly predicting pixel movement. For each pixel in the previous frame, the network outputs a distribution over locations that a pixel is predicted to move. The possible movement a pixel can make are then averaged to obtain the final prediction. The network is trained end-to-end to minimize L2 loss. Mathieu et al. (2016) proposed adversarial training with multiscale convolutional networks to generate sharper pixel-level predictions in comparison to the conventional L2 loss. Villegas et al. (2017b) proposed a network that decomposes motion and content in video prediction and obtained more accurate results over Mathieu et al. (2016). Lotter et al. (2017) proposed a deep predictive coding network in which each layer learns to predict the lower-level difference between the future frame and current frame. As an alternative approach to convolutional encoder-decoder networks, Kalchbrenner et al. (2017) proposed an autoregressive generation scheme for improved prediction performance. In a concurrent work, Babaeizadeh et al. (2018) and Denton and Fergus (2018) proposed stochastic video prediction method based on recurrent variational autoencoders. Despite these efforts, long-term prediction on high-resolution natural videos beyond approximately 20 frames has been known to be very challenging.

Long-term prediction. Oh et al. (2015) proposed an action conditional convolutional encoder-decoder architecture that demonstrated high-quality long-term prediction performance on video games (e.g., Atari games), but it has not been applied to real-world video prediction. Villegas et al. (2017b) proposed a long-term prediction method using a hierarchical approach, but it requires the ground truth landmarks as supervision. Our work proposes several techniques to address this limitation.

4.3 Background

The hierarchical video prediction model in Villegas et al. (2017b) relieves the blurring problem observed in previous prediction approaches by modeling the video dynamics in high-level feature space. This approach enables the prediction of many frames into the future. The hierarchical prediction model is described below.

To predict the image at timestep t, the following procedure is used: First, the high-level features $p_t \in \mathbb{R}^l$ — in this case human pose landmarks — are estimated from the first C context frames. Next, an LSTM is used to predict the future landmark states $\hat{p}_t \in \mathbb{R}^l$ given the landmarks estimated from the context frames as follows:

$$\begin{cases} [\hat{p}_t, H_t] = LSTM(p_{t-1}, H_{t-1}) & \text{if } t \leq C, \\ [\hat{p}_t, H_t] = LSTM(\hat{p}_{t-1}, H_{t-1}) & \text{if } t > C, \end{cases}$$

where $H_t \in \mathbb{R}^h$ is the hidden state of the LSTM at timestep t. Note that the predicted $\hat{p}_t$ after C timesteps is used to generate the video frames. Additionally, they remove the auto-regressive connections that feed $\hat{p}_{t-1}$ back into LSTM making the prediction only depend on H_{t-1}. In our formulation, however, the prediction depends on both $\hat{p}_{t-1}$ and H_{t-1}, but $\hat{p}_{t-1}$ is not a vector of landmarks.

Once all $\hat{p}_t$ are obtained, the visual analogy network (VAN) (Reed et al., 2015b) generates the corresponding image at time t. VAN identifies the change between $g(p_C)$ and $g(\hat{p}_t)$, where $g(.)$ is a fixed function that takes in landmarks and converts them into Gaussian heatmaps. Next, it applies the identified difference to image I_C to generate image I_t. The VAN does this by mapping images to a space where analogies can be represented by additions and subtractions. Therefore, the image at timestep t is computed by

$$\hat{I}_t = VAN(p_C, \hat{p}_t, I_C) =$$
$$f_{dec}(\, f_{pose}(g(\hat{p}_t)) - f_{pose}(g(p_C)) + f_{img}(I_C)\,).$$

In contrast to Villegas et al. (2017b), our method does not require landmarks p_t, and therefore the dependence on the fixed function $g(.)$ is removed. Our method automatically discovers the features needed as input to the VAN for generating frame at time t. These features locate the object moving through time, and help our network focus on generating the moving object pixels in future frames. In the following section, we describe our method and training variations for unsupervised future frame prediction.

4.4 Method

4.4.1 Network Architecture

Our method uses a network architecture similar to Villegas et al. (2017b). However, our predictor LSTM and VAN do not require landmark annotations and can be trained jointly.

In our model, the predictor LSTM is defined by

$$\begin{cases} [\hat{e}_t, H_t] = LSTM(e_{t-1}, H_{t-1}) & \text{if } t \leq C \\ [\hat{e}_t, H_t] = LSTM(\hat{e}_{t-1}, H_{t-1}) & \text{if } t > C, \end{cases} \tag{4.1}$$

where $e_{t-1} \in \mathbb{R}^d$ is a general feature vector computed from an input image I_t by an encoder network, and $\hat{e}_t \in \mathbb{R}^d$ is the feature vector predicted by the LSTM. To compute the frame at time t, we use a variation of the *deep* version of the image analogy formulation from Reed et al. (2015b). In contrast to Villegas et al. (2017b), we use the first frame in the input video to compute the future frames via image analogy. Therefore, the frame at time t is computed by

$$\begin{aligned} \bar{I}_t, M_t = \; & VAN(e_1, \hat{e}_t, I_1) = \\ & f_{dec}(\, f_{enc}(\hat{e}_t) + T(f_{img}(I_1), f_{enc}(e_1), f_{enc}(\hat{e}_t))\,), \end{aligned} \tag{4.2}$$

$$\hat{I}_t = \bar{I}_t \odot M_t + (1 - M_t) \odot I_1, \tag{4.3}$$

where $f_{enc} : \mathbb{R}^d \rightarrow \mathbb{R}^{s \times s \times m}$ is a convolutional network that maps a feature vector into a feature tensor, $f_{img} : \mathbb{R}^{h \times w \times c} \rightarrow \mathbb{R}^{s \times s \times m}$ is a convolutional network that maps an input image into a feature tensor, $f_{dec} : \mathbb{R}^{s \times s \times m} \rightarrow \mathbb{R}^{h \times w \times c}$ is a deconvolutional network that maps a feature tensor into an image, and $T(.,.,.)$ is defined as follows:

$$T(x, y, z) = f_{analogy}([f_{diff}(x - y), z]), \tag{4.4}$$

where $f_{diff} : \mathbb{R}^{s \times s \times m} \rightarrow \mathbb{R}^{s \times s \times m}$ computes a feature tensor from the difference between x and y, $[.,.]$ denotes a concatenation along the depth dimension of the input tensors, and $f_{analogy} : \mathbb{R}^{s \times s \times 2m} \rightarrow \mathbb{R}^{s \times s \times m}$ computes the analogy feature tensor to be added to $f_{enc}(\hat{e}_t)$. Finally, M_t is a gating mechanism that enables our network to identify the moving objects in the video frames. In Equation 4.3, our network chooses pixels from the input frame that can simply be copied into the predicted frame, and pixels that need to be generated are chosen from $\bar{I}_t$. In Section 4.5, we show that the selected areas resemble the structure of moving objects in the input and the predicted frames.

4.4.2 Training Objective

These networks can be trained in multiple ways. In Villegas et al. (2017b), the predictor LSTM and VAN are trained separately using ground truth landmarks. In this work, we explore alternative ways of training these networks in the absence of ground truth annotations of high-level structures.

4.4.2.1 End-to-End Prediction

One simple baseline method is to simply connect the VAN and the predictor LSTM together, and train them end-to-end (E2E). Our full network is optimized to minimize the L2 loss between the predicted image and the ground truth by:

$$\min\left(\sum_{t=1}^{T} L_2(\hat{I}_t, I_t)\right).$$

Figure 4.1 illustrates a diagram of this training scheme. Although a straightforward objective function is optimized, minimizing the L2 loss directly on the image outputs from previous observations tends to produce blurry predictions. This phenomenon has also been observed in several previous works Mathieu et al. (2016); Villegas et al. (2017b,a).

4.4.2.2 Encoder Predictor with Analogy Making

An alternative way to train our network is to constrain the features predicted by LSTM to be close to the outputs of the feature encoder (i.e. $\hat{e}_t \approx e_t$). Simultaneously, the feature encoder outputs can be trained to be useful for analogy making. To accomplish this, we optimize the following objective function:

$$\min\left(\sum_{t=1}^{T} L_2(\hat{I}_t, I_t) + \alpha L_2(\hat{e}_t, e_t)\right), \tag{4.5}$$

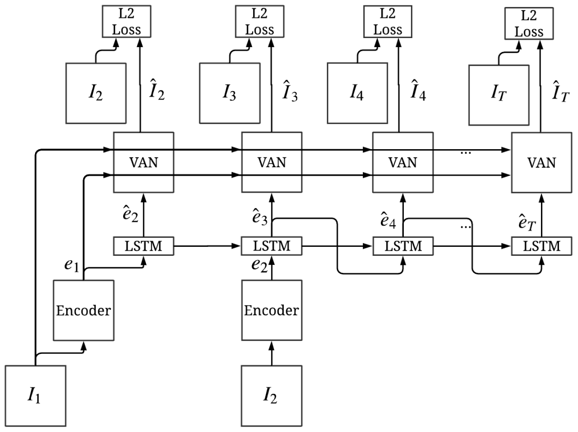


Figure 4.1: The E2E method. The first few frames are encoded and fed into the predictor as context. The predictor predicts the subsequent encodings, which the VAN uses to produce the pixel-level predictions. The average of the losses is minimized. This is the configuration of every method at inference time, even if the predictor and VAN are trained separately.

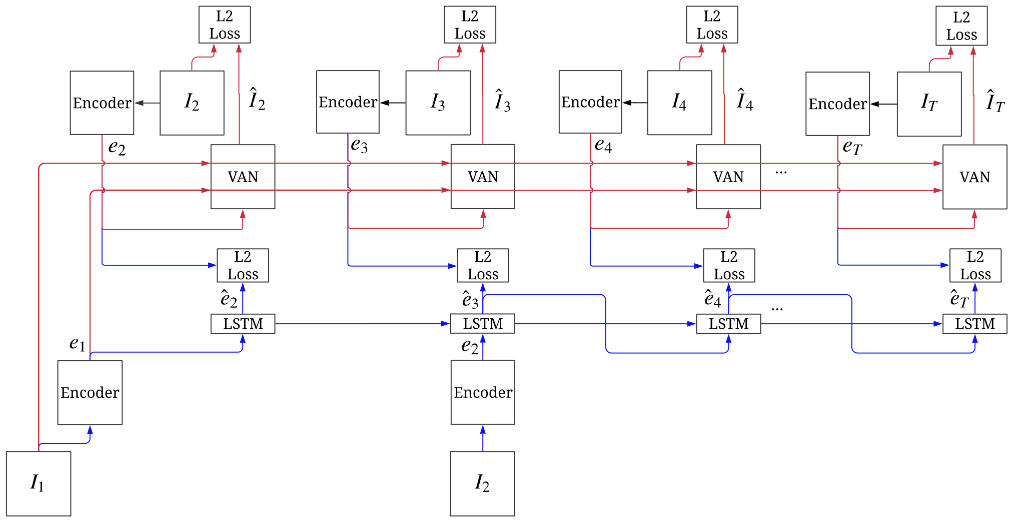


Figure 4.2: Blue lines represent the segment of the EPVA method in which the encoder and predictor are trained together. The encoder is trained to produce an encoding that is easy to predict, and the predictor is trained to predict that encoding into the future. Red lines represent the segment of the EPVA method in which the encoder and the VAN are trained together. The encoder is trained to produce an encoding that is informative to the VAN, while the VAN is trained to output the image given the encoding. The average of the losses in the diagram is minimized. This part of the method is similar to an autoencoder. Our method code is available at `https://bit.ly/2HqiHqx`

where $\hat{I}_t = VAN(e_1, e_t, I_1)$, e_t and e_1 are both outputs of the feature encoder computed from the image at time t and the first image in the video, and α is a balancing hyper parameter that controls the importance between predicting $\hat{e}_t$ that is close to e_t and learning an encoding e_t that is good enough for image analogy. α is used to prevent the predictor and encoder from both outputting the zero feature vector.

Table 4.1: Crowd-sourced human preference evaluation on the moving shapes dataset.

Method	Shape has correct color	Shape has wrong color	Shape disappeared
EPVA	96.9%	3.1%	0%
CDNA Baseline	24.6%	5.7%	69.7%

Figure 4.2 illustrates the flow of information by which the encoder and predictor are trained together with blue arrows, and the flow of information by which the VAN and encoder are trained together with red arrows. Separate gradient descent procedures (or optimizers, in TensorFlow parlance) could be used to minimize $L_2(\hat{I}_t, I_t)$ and $L_2(\hat{e}_t, e_t)$, but we found that minimizing the sum is more accurate in our experiments. With this method, the predictor will generate the encoder outputs in future time steps, and the VAN will use the encoder output to produce the frame. The advantage of this training scheme is that the VAN learns to sharply predict the pixels since it is trained given the encoding from the ground truth frame. The predictor learns to approximate the ground truth high-level features from the encoder. Therefore, at inference time the VAN knows how to decode the high-level structure features resulting in better predictions. Note that the encoder outputs e_t are given to VAN as input during training; however, the predictor outputs $\hat{e}_t$ are given during testing. We refer to this method as EPVA.

The EPVA method works most accurately when experimented with α starting small, around 1e-7, and gradually increased to around 0.1 during training. As a result, the encoder will first be optimized to produce an informative encoding, then gradually optimized to make that encoding easy to predict by the predictor.

4.4.2.3 EPVA with adversarial loss in predictor

A disadvantage of the EPVA training scheme alone is that the predictor is trained to minimize the L2 loss with respect to the encoder outputs. The L2 loss is notoriously known for the "blurriness effect," and it causes our predictor LSTM to output blurry predictions in encoding space.

One solution to this problem is to use an adversarial loss Goodfellow et al. (2014b)

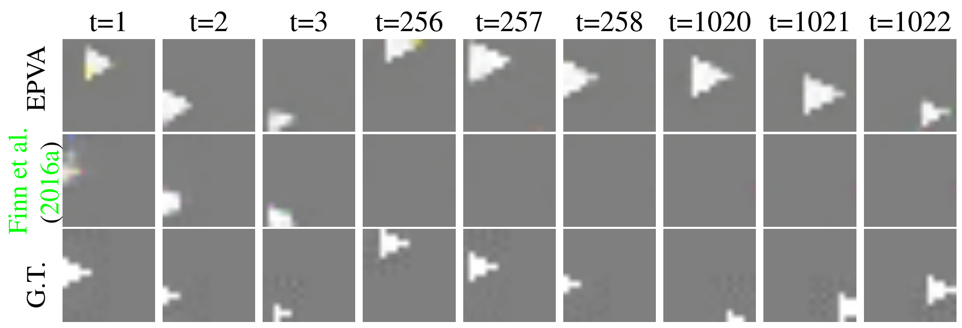


Figure 4.3: A visual comparison of the EPVA method and CDNA from Finn et al. (2016a) as the baseline. This is a representative example of the quality of predictions from both methods. For videos please visit `https://bit.ly/2kS8r16`.

between the predictor and encoder. We use an LSTM discriminator network, which takes a sequence of encodings and produces a score that indicates whether the encodings came from the predictor or the encoder network. We train the discriminator to minimize the improved Wasserstein loss Gulrajani et al. (2017).

$$\min\Big(\sum_{t=1}^{T}(D(\hat{e}) - D(e) + \lambda(\|\nabla_{\hat{e}} D(\hat{e})\|_2 - 1)^2])\Big). \tag{4.6}$$

Here, e and $\hat{e}$ are the sequence of inferred and predicted encodings respectively. We train both the encoder and the predictor, so we use a loss which takes both the encoder and predictor outputs into account. Therefore, we use the negative of the discriminator loss to optimize the generator.

$$\min\Big(-\sum_{t=1}^{T}(D(\hat{e}) - D(e))\Big) \tag{4.7}$$

We also still optimize the l2 loss between the predictor and encoder, weighted by a scale factor. This ensures the predictions will be accurate given the context frame. We also feed a Gaussian noise variable into the predictor in order to generate different results given the same input sequence. We found that the noise helps generate more complex predictions in practice.

In addition to passing the predictor or encoder output to the discriminator, we also pass the output of the VAN encoder, given the predictor or encoder output. This trains the predictor and encoder to encourage the VAN to produce similar quality images. This is achieved by substituting $[f_{enc}(e), e]$ for e and $[f_{enc}(\hat{e}), \hat{e}]$ for $\hat{e}$ in the equations above, where f_{enc} is the VAN encoder. The encoder and VAN are trained together in the same way as previously discussed.

4.5 Experiments

We evaluated our methods on two datasets: the Human 3.6M dataset (Ionescu et al., 2014b, 2011), and a toy dataset based on videos of bouncing shapes. More sample videos and code to reproduce our results are available at our project website `https://bit.ly/2kS8r16`.

4.5.1 Long-term Prediction on a Toy Dataset

We train our method on a toy task with known factors of variation. We used a dataset with a generated shape that bounces around the image and changes size deterministically. We trained our EPVA method and the CDNA method from Finn et al. (2016a) to predict 16 frames, given the first three frames as context. Both methods are evaluated on predicting approximately 1000 frames. We added noise to the LSTM states of the predictor network during training to help predict accurate motion further into the future. Results from a held out test set are described in the following.

After visually analyzing the results of both methods, we found that when the CDNA fails, the shape disappears entirely. In contrast, when the EPVA method fails, the shape changes color. See Figure 4.3 for sample predictions. For quantitative evaluation, we used a script to measure whether a shape was present from frames 1012 to 1022 and if that shape has the appropriate color. Table 4.1 shows the results averaged over 1000 runs. The CDNA method predicts a shape with the correct color about 25% of the time, and the EPVA method predicts a shape with the correct color about 97% of the time. The EPVA method sometimes fails by predicting the shape in the same location from frame to frame, but this is rare as the reader can confirm by examining the randomly sampled predictions on our project website. It is unrealistic to expect the methods to predict the location of the shape accurately in frame

Table 4.2: Crowd-sourced human preference evaluation on the Human 3.6M dataset.

Comparison	**Ours is better**	**Same**	**Baseline is better**
EPVA ADV. 1-127 vs Finn et al. (2016a) 1-127	73.9%	13.2%	12.9%
EPVA ADV. 5-127 vs Denton and Fergus (2018) 5-127	58.2%	24.0%	17.8%

1000 since small errors propagate in each prediction step.

4.5.2 Long-term Prediction on Human 3.6M

In these experiments, we use subjects 1, 5, 6, 7, and 8 for training, and subject 9 for validation. Subject 11 results are reported in this paper for testing. We use 64 by 64 images, and subsample the dataset to 6.25 frames per second. We train the methods to predict 32 frames and the results in this paper show predictions over 126 frames. Each method is given the first five frames as context. In these images, the model predicts about 20 seconds into the future starting with 0.8 seconds of context. We use an encoding dimension of 64 for variations of our method on this dataset. The encoder in the EPVA method is initialized with the VGG network Simonyan and Zisserman (2015b) pretrained on Imagenet (Deng et al., 2009). To speed up the convergence of the EPVA ADVERSARIAL method, we start training from a pretrained EPVA model.

We compare our method to the CDNA method in Finn et al. (2016a) and the SVG-LP method in Denton and Fergus (2018). We trained each method with the same number of frames and context frames as ours. For Denton and Fergus (2018), we performed grid search on the β and learning rate to find the best configuration for this experiment, as well as, used a network as large as we could fit in the GPU. For Finn et al. (2016a), we performed grid search on the learning rate. The method in Denton and Fergus (2018) can predict multiple futures, so we generate 5 futures for each context sequence, and compare against the one that most closely matches the ground truth in terms of SSIM. We find that this produces slightly better results than taking random predictions. Note that this protocol provides an unfair advantage to their method.

Figure 4.5 shows comparison to the baselines, and different variations of our method are compared in Figure 4.6. In Figure 4.5, we also show the discovered foreground motion segmentation mask from our method. This mask clearly shows that the feature embeddings from our encoder and predictor encode the rough location and outline of the moving human.

From visually analyzing the results, we found that the E2E and CDNA methods usually blur out very quickly. The EPVA method produces accurate predictions further into the future, but the figure sometimes disappears. The human predictions from the EPVA ADVERSARIAL method disappear less often and usually reappear in a later time step.

The CDNA Finn et al. (2016a) and the E2E methods produce blurry images because they are trained to minimize L2 loss directly. In the EPVA method, the predictor and VAN are trained separately. This prevents the VAN from learning to produce blurry images when the predictor is not confident. The predictions will be sharp as long as the predictor network outputs a valid encoding. The EPVA ADVERSARIAL method makes the predictor network more likely to produce a valid encoding since the discriminator is trained to produce valid predictions. We also observe that there is more movement in the EPVA ADVERSARIAL method.

4.5.2.1 Person Detector Evaluation

We propose to compare the methods quantitatively by considering whether the generated videos contain a recognizable person. To do this in an automated fashion, we ran a MobileNet (Howard et al., 2017) object detection model pretrained on the MS-COCO (Lin et al., 2014) dataset for each of the generated frames. We record the confidence of the detector that a person (one of the MS-COCO labels) is in the image. We call this the "person score" (with value ranges from 0 to 1, with a higher score corresponding to a higher confidence level). The human detector achieves approximately an accuracy of 0.4 on the ground truth data. The results on each frame averaged over 1000 runs are shown in Figure 4.4. The EPVA ADVERSARIAL method stays relatively constant over the different frames. For longer-term predictions, the evaluation shows that the EPVA ADVERSARIAL method is significantly better than the baselines.

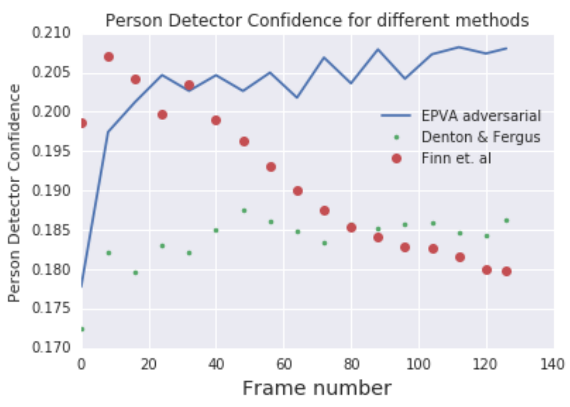


Figure 4.4: Confidence of the person detector that a person is recognized in the predicted frame ("person score").

4.5.2.2 Human Evaluation

We also use a service similar to Mechanical Turk to collect comparisons of 1,000 generated videos from Finn et al. (2016a) and Denton and Fergus (2018) to different variations of our method. The task presents videos generated by the two methods side by side to human raters and asks them to confirm whether one of the videos is more realistic. The instructions tell raters to look for realistic motion, as well as a realistic person image. To evaluate the quality of the long-term predictions from the EPVA ADVERSARIAL method, we compare frames 64 to 127 of the EPVA ADVERSARIAL method to frames 1 to 63 of Finn et al. (2016a). We evaluate frames 5-127 of Denton and Fergus (2018) against 5-127 of ours since their method isn't designed to produce good results for the context frames.

The summary results are shown in Table 4.2. From these results, we conclude the following: the EPVA method generates significantly better long-term predictions than Finn et al. (2016a). Further, the EPVA ADVERSARIAL method is a dramatic improvement over the EPVA method. The EPVA ADVERSARIAL method is capable of high-quality long-term predictions, as shown by frames 64 to 127 (seconds 10 to 20) of the EPVA ADVERSARIAL method being rated higher than frames 1-63 of Finn et al. (2016a). The EPVA ADVERSARIAL is also significantly better than Denton and Fergus (2018) even after choosing the best out of 5 predictions after comparing with the ground truth in terms of

SSIM.

4.5.2.3 Pose regression from learned features

We perform experiments using the learned encoder features for human pose regression. We compare against a baseline based on features computed using the VGG network Simonyan and Zisserman (2015b) trained for object recognition. The features are used as input to a 2-layer MLP, and trained to output human pose landmarks. The MLP trained with our features achieves an error of 0.0687 against an error of 0.0758 from the baseline features. This is a relative improvement of approximately 9%. This along with the generated masks shows the usefulness of our discovered features.

4.5.3 Ablation Studies

We perform the following experiments to test different variations of the network and training. We hypothesize that using a VAN improves the quality of the predictions. To test this, we train a version of the network with the VAN replaced by a decoder network that only had access to the encoding and not the first observed frame.

In this method, as well as the methods with the VAN, the decoder outputs a mask that controls whether to use its own output, or the pixels of the first frame. Thus, the decoder will have to set the mask values to not use the pixels from the first frame that correspond to the image of the person. Without the VAN, the network is often unable to set the mask values to completely remove the human from the first frame when predicting frames beyond 32. This is because the network is not always given access to the first frame, so it has to represent both foreground and background information in the prediction, which degrades over time. Refer to Figure 4.6 for comparison.

We also tried to use a hybrid objective that combines E2E and EPVA losses, but the videos generated from this method are more blurry than the videos from the EPVA method. These are called E2E and EPVA in Figure 4.6. Finally, we also trained and evaluated the EPVA method with 10 frames of context instead of 5. We found that this didn't improve the long-term prediction results.

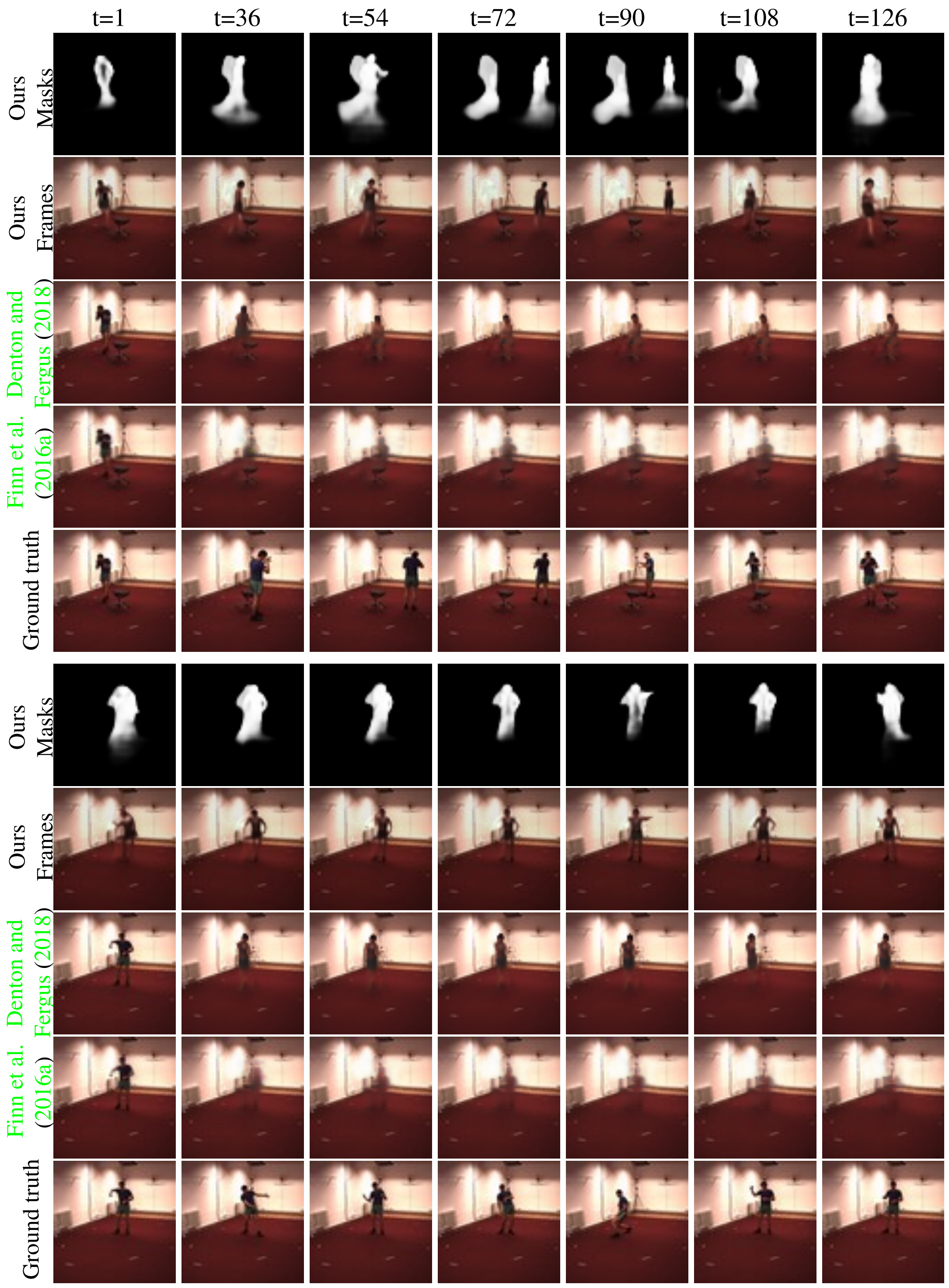


Figure 4.5: Comparison of the generated videos from EPVA with the ADVERSARIAL LOSS (ours), CDNA Finn et al. (2016a), and SVG-LP Denton and Fergus (2018). We let each method predict 127 frames and show the time steps indicated on top of the figure. The person completely disappears in all the predictions generated using Finn et al. (2016a). For the SVG-LP method Denton and Fergus (2018), the person either stops moving or almost vanishes into the background. The EPVA with ADVERSARIAL LOSS method produces sharp predictions in comparison to the baselines. Additionally, we show the discovered foreground motion segmentation mask that allows our network to delete the human in the input frame (static mask in the top example) and generate the human in the future frames (moving mask in the top example). Please refer to our project website for video results: `https://bit.ly/2kS8r16`.

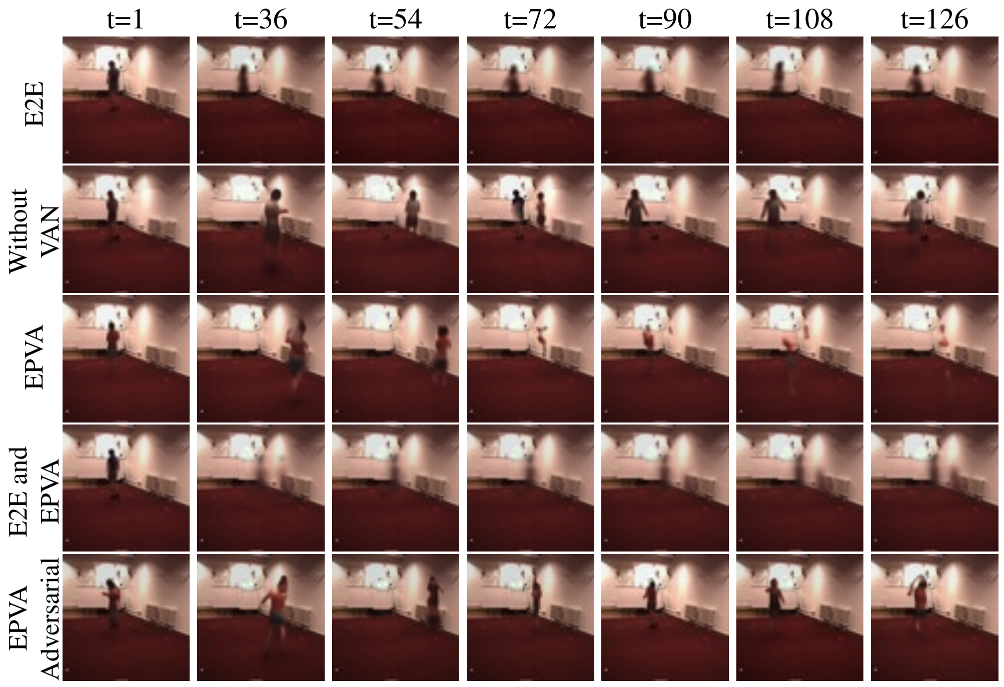


Figure 4.6: Ablative study illustration. We present comparisons between different variations of our architecture: E2E, loss without VAN, EPVA, combined E2E and EPVA loss, and our best model configuration (EPVA ADVERSARIAL). See our project website for videos.

4.6 Conclusion

We presented hierarchical long-term video prediction approaches that do not require ground truth high-level structure annotations. The proposed EPVA method has the limitation of the predictions occasionally disappearing, but it generates sharper images for a longer period of time compared to Finn et al. (2016a), and the E2E method. By applying adversarial loss in the higher-level feature space, our EPVA ADVERSARIAL method generates more realistic predictions compared to all of the presented baselines including Finn et al. (2016a) and Denton and Fergus (2018). This result suggests that it is beneficial to apply an adversarial loss in the higher-level feature space. For future work, applying other techniques in feature space such as the variational method described in Babaeizadeh et al. (2018) could enable our network to generate multiple future trajectories.

Acknowledgments. We thank colleagues at Google Brain and anonymous reviewers for their constructive feedback and suggestions about this work. We also thank Emily Denton for providing her code available for comparison. R. Villegas was supported by Rackham

Merit Fellowship.

CHAPTER V

High Fidelity Video Prediction with Large Neural Nets

Predicting future video frames is extremely challenging, as there are many factors of variation that make up the dynamics of how frames change through time. Previously proposed solutions require complex inductive biases inside network architectures with highly specialized computation, including segmentation masks, optical flow, and foreground and background separation. In this work, we question if such handcrafted architectures are necessary and instead propose a different approach: finding minimal inductive bias for video prediction while maximizing network capacity. We investigate this question by performing the first large-scale empirical study and demonstrate state-of-the-art performance by learning large models on three different datasets: one for modeling object interactions, one for modeling human motion, and one for modeling car driving.

5.1 Introduction

From throwing a ball to driving a car, humans are very good at being able to interact with objects in the world and anticipate the results of their actions. Being able to teach agents to do the same has enormous possibilities for training intelligent agents capable of generalizing to many tasks. Model-based reinforcement learning is one such technique that seeks to do this – by first learning a model of the world, and then by planning with the learned model.

There has been some recent success with training agents in this manner by first using video prediction to model the world. Particularly, video prediction models combined with simple planning algorithms (Hafner et al., 2019) or policy-based learning (Kaiser et al., 2019) for model-based reinforcement learning have been shown to perform equally or better than model-free methods with far less interactions with the environment. Additionally, Ebert et al. (2018) showed that video prediction methods are also useful for robotic control, especially with regards to specifying unstructured goal positions.

However, training an agent to accurately predict what will happen next is still an open problem. Video prediction, the task of generating future frames given context frames, is notoriously hard. There are many spatio-temporal factors of variation present in videos that make this problem very difficult for neural networks to model. Many methods have been proposed to tackle this problem (Oh et al., 2015; Finn et al., 2016b; Vondrick et al., 2016; Villegas et al., 2017a; Lotter et al., 2017; Denton and Birodkar, 2017; Tulyakov et al., 2018; Liang et al., 2017; Denton and Fergus, 2018; Wichers et al., 2018; Babaeizadeh et al., 2018; Lee et al., 2018; Byeon et al., 2018; Yan et al., 2018; Kumar et al., 2018). Most of these works propose some type of separation of information streams (e.g., motion/pose and content streams), specialized computations (e.g., warping, optical flow, foreground/background masks, predictive coding, etc), additional high-level information (e.g., landmarks, semantic segmentation masks, etc) or are simply shown to work in relatively simpler environments (e.g., Atari, synthetic shapes, centered human faces and bodies, etc).

Simply making neural networks larger has been shown to improve performance in many areas such as image classification (Real et al., 2018; Zoph et al., 2018; Huang et al., 2018), image generation (Brock et al., 2019), and language understanding (Devlin et al., 2018; Radford et al., 2019), amongst others. Particularly, Brock et al. (2019) recently showed that increasing the capacity of GANs (Goodfellow et al., 2014a) results in dramatic improvements for image generation.

In his blog post "The Bitter Lesson", Rich Sutton comments on these types of developments by arguing that the most significant breakthroughs in machine learning have come from increasing the compute provided to simple models, rather than from specialized, hand-crafted architectures (Sutton, 2019). For example, he explains that the early specialized algorithms of computer vision (edge detection, SIFT features, etc.) gave way to larger but simpler convolutional neural networks. In this work, we seek to answer a similar question: do we really need specialized architectures for video prediction? Or is it sufficient to maximize network capacity on models with minimal inductive bias?

In this work, we perform the first large-scale empirical study of the effects of minimal inductive bias and maximal capacity on video prediction. We show that without the need of optical flow, segmentation masks, adversarial losses, landmarks, or any other forms of inductive bias, it is possible to generate high quality video by simply increasing the scale of computation. Overall, our experiments demonstrate that: (1) large models with minimal inductive bias tend to improve the performance both qualitatively and quantitatively, (2) recurrent models outperform non-recurrent models, and (3) stochastic models perform better than non-stochastic models, especially in the presence of uncertainty (e.g., videos with unknown action or control).

5.2 Related Work

The task of predicting multiple frames into the future has been studied for a few years now. Initially, many early methods tried to simply predict future frames in small videos or patches from large videos (Michalski et al., 2014b; Ranzato et al., 2014; Srivastava et al., 2015). This type of video prediction caused rectangular-shaped artifacts when attempting to fuse the predicted patches, since each predicted patch was blind to its surroundings. Then, action-conditioned video prediction models were built with the aim of being used for model-based reinforcement learning (Oh et al., 2015; Finn et al., 2016b). Later, video prediction models started becoming more complex and better at predicting future frames. Lotter et al. (2017) proposed a neural network based on predictive coding. Villegas et al.

(2017a) proposed to separate motion and content streams in video input. Villegas et al. (2017b) proposed to predict future video as landmarks in the future and then use these landmarks to generate frames. Denton and Birodkar (2017) proposed to have a pose and content encoders as separate information streams. However, all of these methods focused on predicting a single future. Unfortunately, real-world video is highly stochastic – that is, there are multiple possible futures given a single past.

Many methods focusing on the stochastic nature of real-world videos have been recently proposed. Babaeizadeh et al. (2018) build on the optical flow method proposed by Finn et al. (2016b) by introducing a variational approach to video prediction where the entire future is encoded into a posterior distribution that is used to sample latent variables. Lee et al. (2018) also build on optical flow and propose an adversarial version of stochastic video prediction where two discriminator networks are used to enable sharper frame prediction. Denton and Fergus (2018) also propose a similar variational approach. In their method, the latent variables are sampled from a prior distribution of the future during inference time, and only frames up to the current time step are used to model the future posterior distribution. Kumar et al. (2018) propose a method based on normalizing flows where the exact log-likelihood can be computed for training.

In this work, we investigate whether we can achieve high quality video predictions without the use of the previously mentioned techniques (optical flows, adversarial objectives, etc.) by just maximizing the capacity of a standard neural network. To the best of our knowledge, this work is the first to perform a thorough investigation on the effect of capacity increases for video prediction.

5.3 Scaling up video prediction

In this section, we present our method for scaling up video prediction networks. We first consider the Stochastic Video Generation (SVG) architecture presented in Denton and Fergus (2018), a stochastic video prediction model that is entirely made up of standard neural network layers without any special computations (e. g. optical flow). SVG is competitive with other state-of-the-art stochastic video prediction models (SAVP, SV2P) (Lee et al., 2018); however, unlike SAVP and SV2P, it does not use optical flow, adversarial losses, etc. As such, SVG was a fitting starting point to our investigation.

To build our baseline model, we start with the stochastic component that models the inherent uncertainty in future predictions from Denton and Fergus (2018). We also use shallower encoder-decoders that only have convolutional layers to enable more detailed image reconstruction (Dosovitskiy and Brox, 2016b). A slightly shallower encoder-decoder architecture results in less information lost in the latent state, as the resulting convolutional map from the bottlenecked layers is larger. Then, in contrast to Denton and Fergus (2018), we use a convolutional LSTM architecture, instead of a fully-connected LSTM, to fit the shallow encoders-decoders. Finally, the last difference is that we optimize the ℓ_1 loss with respect to the ground-truth frame for all models like in the SAVP model, instead of using ℓ_2 like in SVG. Lee et al. (2018) showed that ℓ_1 encouraged sharper frame prediction over ℓ_2.

We optimize our baseline architecture by maximizing the following variational lower-bound:

$$\sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})} \log p_\theta(\mathbf{x}_t|\mathbf{z}_{\leq t}, \mathbf{x}_{<t}) \; - \beta D_{KL}\left(q_\phi(\mathbf{z}_t|\mathbf{x}_{\leq t}) || p_\psi(\mathbf{z}_t|\mathbf{x}_{<t})\right),$$

where $\mathbf{x}_t$ is the frame at time step t, $q_\phi(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})$ the approximate posterior distribution, $p_\psi(\mathbf{z}_t|\mathbf{x}_{<t})$ is the prior distribution, $p_\theta(\mathbf{x}_t|\mathbf{z}_{\leq t}, \mathbf{x}_{<t})$ is the generative distribution, and β regulates the strength of the KL term in the lowerbound. During training time, the frame

prediction process at time step t is as follows:

$$\begin{aligned}
\mu_\phi(t), \sigma_\phi(t) &= \text{LSTM}_\phi(\mathbf{h}_t; M) && \text{where} \quad \mathbf{h}_t = f^{\text{enc}}\left(\mathbf{x}_t; K\right), \\
\mathbf{z}_t &\sim \mathcal{N}\left(\mu_\phi(t), \sigma_\phi(t)\right), \\
\mathbf{g}_t &= \text{LSTM}_\theta\left(\mathbf{h}_{t-1}, \mathbf{z}_t; M\right) && \text{where} \quad \mathbf{h}_{t-1} = f^{\text{enc}}\left(\mathbf{x}_{t-1}; K\right), \\
\mathbf{x}_t &= f^{\text{dec}}\left(\mathbf{g}_t; K\right),
\end{aligned}$$

where f^{enc} is an image encoder and f^{dec} is an image decoder neural network. LSTM$_\phi$ and LSTM$_\theta$ are LSTMs modeling the posterior and generative distributions, respectively. $\mu_\phi(t)$ and $\sigma_\phi(t)$ are the parameters of the posterior distribution modeling the Gaussian latent code $\mathbf{z}_t$. Finally, $\mathbf{x}_t$ is the predicted frame at time step t.

To increase the capacity of our baseline model, we use hyperparameters K and M, which denote the factors by which the number of neurons in each layer of the encoder, decoder and LSTMs are increased. For example, if the number of neurons in LSTM is d, then we scale up by $d \times M$. The same applies to the encoder and decoder networks but using K as the factor. In our experiments we increase both K and M together until we reach the device limits. Due to the LSTM having more parameters, we stop increasing the capacity of the LSTM at $M = 3$ but continue to increase K up to 5. At test time, the same process is followed, however, the posterior distribution is replaced by the Gaussian parameters computed by the prior distribution:

$$\mu_\psi(t), \sigma_\psi(t) = \text{LSTM}_\psi(\mathbf{h}_{t-1}; M) \qquad \text{where} \quad \mathbf{h}_{t-1} = f^{\text{enc}}\left(\mathbf{x}_{t-1}; K\right),$$

Next, we perform ablative studies on our baseline architecture to better quantify exactly how much each individual component affects the quality of video prediction as capacity increases. First, we remove the stochastic component, leaving behind a fully deterministic architecture with just a CNN-based encoder-decoder and a convolutional LSTM. For this version, we simply disable the prior and posterior networks as described above. Finally, we remove the LSTM component, leaving behind only the encoder-decoder CNN architectures. For this version, we simply use f^{enc} and f^{dec} as the full video prediction network. However, we let f^{enc} observe the same number of initial history as the recurrent counterparts.

Details of the devices we use to scale up computation can be found in the supplementary material.

5.4 Experiments

In this section, we evaluate our method on three different datasets, each with different challenges.

Object interactions. We use the action-conditioned towel pick dataset from Ebert et al. (2018) to evaluate how our models perform with standard object interactions. This dataset contains a robot arm that is interacting with towel objects. Even though this dataset uses action-conditioning, stochastic video prediction is still required for this task. This is because the motion of the objects is not fully determined by the actions (the movements of the robot arm), but also includes factors such as friction and the objects' current state. For this dataset, we resize the original resolution of 48x64 to 64x64. For evaluation, we use the first 256 videos in the test set as defined by Ebert et al. (2018).

Structured motion. We use the Human 3.6M dataset (Ionescu et al., 2014a) to measure the ability of our models to predict structured motion. This dataset is comprised of humans performing actions inside a room (walking around, sitting on a chair, etc.). Human motion is highly structured (i.e., many degrees of freedom), and so, it is difficult to model. We use the train/test split from Villegas et al. (2017b). For this dataset, we resize the original resolution of 1000x1000 to 64x64.

Partial observability. We use the KITTI driving dataset (Geiger et al., 2013) to measure how our models perform in conditions of partial observability. This dataset contains driving scenes taken from a front camera view of a car driving in the city, residential neighborhoods,

	CNN models		LSTM models		SVG' models	
Dataset	Biggest (M=3, K=5)	Baseline (M=1, K=1)	Biggest (M=3, K=5)	Baseline (M=1, K=1)	Biggest (M=3, K=5)	Baseline (M=1, K=1)
Towel Pick	199.81	281.07	100.04	206.49	**93.71**	189.91
Human 3.6M	1321.23	1077.55	458.77	614.21	**429.88**	682.08
KITTI	2414.64	2906.71	**1159.25**	2502.69	1217.25	2264.91

Table 5.1: **Fréchet Video Distance evaluation (lower is better)**. We compare the biggest model we were able to train against the baseline models (M=1, K=1). Note that all models (SVG', CNN, and LSTM). The biggest recurrent models are significantly better than their small counterpart. Please refer to our supplementary material for plots showing how gradually increasing model capacity results in better performance.

and on the road. The front view camera of the vehicle causes partial observability of the vehicle environment, which requires a model to generate seen and unseen areas when predicting future frames. We use the train/test split from Lotter et al. (2017) in our experiments. We extract 30 frame clips and skip every 5 frames from the test set so that the test videos do not significantly overlap, which gives us 148 test clips in the end. For this dataset, we resize the original resolution of 128x160 to 64x64.

5.4.1 Evaluation metrics

We perform a rigorous evaluation using five different metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM), VGG Cosine Similarity, Fréchet Video Distance (FVD) (Unterthiner et al., 2018), and human evaluation from Amazon Mechanical Turk (AMT) workers. We perform these evaluations on all models described in Section 5.3: our baseline (denoted as SVG'), the recurrent deterministic model (denoted as LSTM), and the encoder-decoder CNN model (denoted as CNN). In addition, we present a study comparing the video prediction performance as a result of using skip-connections from every layer of the encoder to every layer of the decoder versus not using skip connections at all (Supplemetary C.0.3).

5.4.1.1 Frame-wise evaluation

We use three different metrics to perform frame-wise evaluation: PSNR, SSIM, and VGG cosine similarity. PSNR and SSIM perform a pixel-wise comparison between the predicted frames and generated frames, effectively measuring if the exact pixels have been generated. VGG Cosine Similarity has been used in prior work (Lee et al., 2018) to compare frames in a perceptual level. VGGnet (Simonyan and Zisserman, 2015c) is used to extract features from the predicted and ground-truth frames, and cosine similarity is performed at feature-level. Similar to Kumar et al. (2018); Babaeizadeh et al. (2018); Lee et al. (2018), we sample 100 future trajectories per video and pick the highest scoring trajectory as the main score.

5.4.1.2 Dynamics-based evaluation

We use two different metrics to measure the overall realism of the generated videos: FVD and human evaluations. FVD, a recently proposed metric for video dynamics accuracy, uses a 3D CNN trained for video classification to extract a single feature vector from a video. Analogous to the well-known FID (Heusel et al., 2017), it compares the distribution of features extracted from ground-truth videos and generated videos. Intuitively, this metric compares the quality of the overall predicted video dynamics with that of the ground-truth videos rather than a per-frame comparison. For FVD, we also sample 100 future trajectories per video, but in contrast, all 100 trajectories are used in this evaluation metric (i.e., not just the max, as we did for VGG cosine similarity).

We also use Amazon Mechanical Turk (AMT) workers to perform human evaluations. The workers are presented with two videos (baseline and largest models) and asked to either select the more realistic video or mark that they look about the same. We choose the videos for both models by selecting the highest scoring videos in terms of the VGG cosine similarity with respect to the ground truth. We use 10 unique workers per video and choose the selection with the most votes as the final answer. Finally, we also show qualitative evaluations on pairs of videos, also selected by using the highest VGG cosine similarity scores for both the baseline and the largest model. We run the human perception based evaluation on the best two architectures we scale up.

	LSTM models			SVG' models		
Dataset	Biggest (M=3, K=5)	Baseline (M=1, K=1)	About the same	Biggest (M=3, K=5)	Baseline (M=1, K=1)	About the same
Towel Pick	**90.2**%	9.0%	0.8%	**68.8**%	25.8%	5.5%
Human 3.6M	**98.7**%	1.3%	0.0%	**95.8**%	3.4%	0.8%
KITTI	**99.3**%	0.7%	0.0%	**99.3**%	0.7%	0.0%

Table 5.2: **Amazon Mechanical Turk human worker preference**. We compared the biggest and baseline models from LSTM and SVG'. The bigger models are more frequently preferred by humans. We present a full comparison for all large models in Supplementary C.0.4.

5.4.2 Robot arm

For this dataset, we perform action-conditioned video prediction. We modify the baseline and large models to take in actions as additional input to the video prediction model. Action conditioning does not take away the inherent stochastic nature of video prediction due to the dynamics of the environment. During training time, the models are conditioned on 2 input frames and predict 10 frames into the future. During test time, the models predict 18 frames into the future.

Dynamics-based evaluation. We first evaluate the action-conditioned video prediction models using FVD to measure the realism in the dynamics. In Table 5.1 (top row), we present the results of scaling up the three models described in Section 5.3. Firstly, we see that our baseline architecture improves dramatically at the largest capacity we were able to train. Secondly, for our ablative experiments, we notice that larger capacity improves the performance of the vanilla CNN architecture. Interestingly, by increasing the capacity of the CNN architecture, it approaches the performance of the baseline SVG' architecture. However, as capacity increases, the lack of recurrence heavily affects the performance of the vanilla CNN architecture in comparison with the models that do have an LSTM (Supplementary C.0.2.1). Both the LSTM model and SVG' perform similarly well, with SVG' model performing slightly better. This makes sense as the deterministic LSTM model is more likely to produce videos closer to the ground truth; however, the stochastic component is still quite important as a good video prediction model must be both realistic

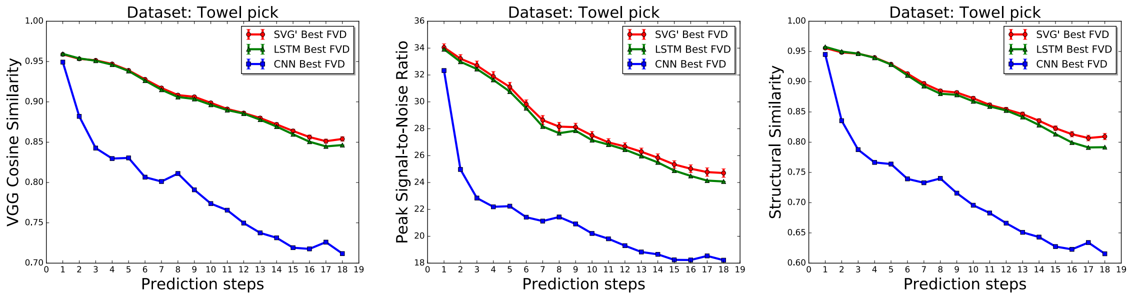


Figure 5.1: Towel pick per-frame evaluation (higher is better). We compare the best performing models in terms of FVD. For model capacity comparisons, please refer to Supplementary C.0.2.1.

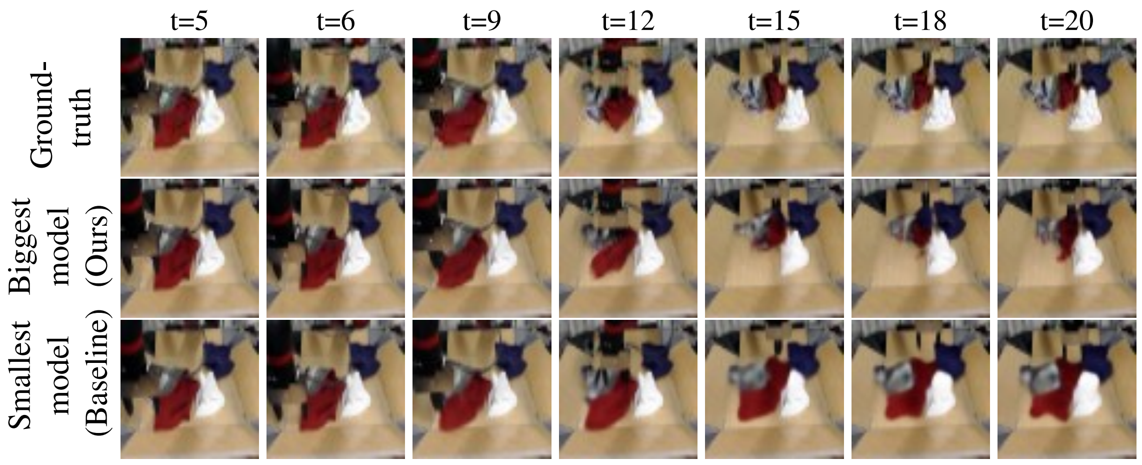


Figure 5.2: Robot towel pick qualitative evaluation. Our highest capacity model (middle row) produces better modeling of the robot arm dynamics, as well, as object interactions. The baseline model (bottom row) fails at modeling the objects (object blurriness), and also, the robot arm dynamics are not well modeled (gripper is open when the it should be close at t=18). For best viewing and more results, please visit our website `https://cutt.ly/QGuCex`.

and capable of handling multiple possible futures. Finally, we use human evaluations through Amazon Mechanical Turk to compare our biggest models with the corresponding baselines. We asked workers to focus on how realistic the interaction between the robot arm and objects looks. As shown in Table 5.2, the largest SVG' is preferred 68.8% of the time vs 25.8% of the time for the baseline (right), and the largest LSTM model is preferred 90.2% of the time vs 9.0% of the time for the baseline (left).

Frame-wise evaluation. Next, we use FVD to select the best models from CNN, LSTM, and SVG', and perform frame-wise evaluation on each of these three models. Since models that copy background pixels perfectly can perform well on these frame-wise evaluation metrics, in the supplementary material we also discuss a comparison against a simple baseline where the last observed frame is copied through time. From Figure 5.1, we can see that the CNN model performs much worse than the models that have recurrent connections. This is a clear indication that recurrence is necessary to predict future frames, and capacity cannot make up for it. Both LSTM and SVG perform similarly well, however, towards the end, SVG slightly outperforms LSTM. The full evaluation on all capacities for SVG', LSTM, and CNN is presented in the supplementary material.

Qualitative evaluation. In Figure 5.2, we show example videos from the smallest SVG' model, the largest SVG' model, and the ground truth. The predictions from the small baseline model are blurrier compared to the largest model, while the edges of objects from the larger model's predictions stay continuously sharp throughout the entire video. This is clear evidence that increasing the model capacity enables more accurate modeling of the pick up dynamics. For more videos, please visit our website `https://cutt.ly/QGuCex`

5.4.3 Human activities

For this dataset, we perform action-free video prediction. We use a single model to predict all action sequences in the Human 3.6M dataset. During training time, the models are conditioned on 5 input frames and predict 10 frames into the future. At test time, the models predict 25 frames.

Dynamics-based evaluation. We evaluate the predicted human motion with FVD (Table 5.1, middle row). The performance of the CNN model is poor in this dataset, and increasing the capacity of the CNN does not lead to any increase in performance. We hypothesize that this is because the lack of action conditioning and the many degrees of freedom in human motion makes it very difficult to model with a simple encoder-decoder CNN. However, after adding recurrence, both LSTM and SVG' perform significantly better, and both models' performance become better as their capacity is increased (Supplementary C.0.2.2). Similar to Section 5.4.2, we see that SVG' performs better than LSTM. This is again likely due to

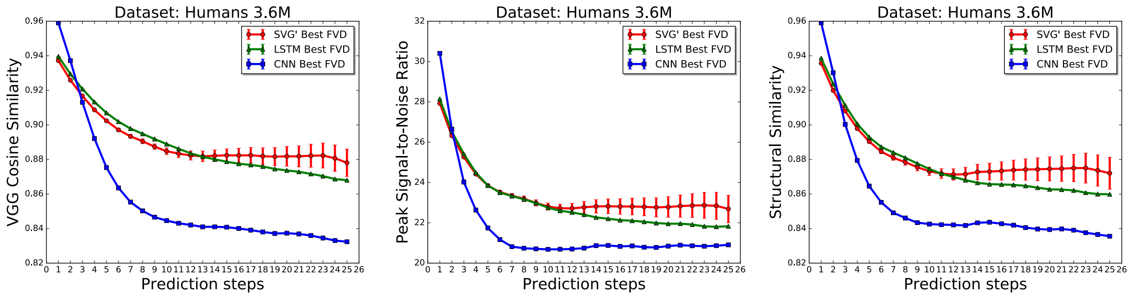


Figure 5.3: Human 3.6M per-frame evaluation (higher is better). We compare the best performing models in terms of FVD. For model capacity comparisons, please refer to Supplementary C.0.2.2.

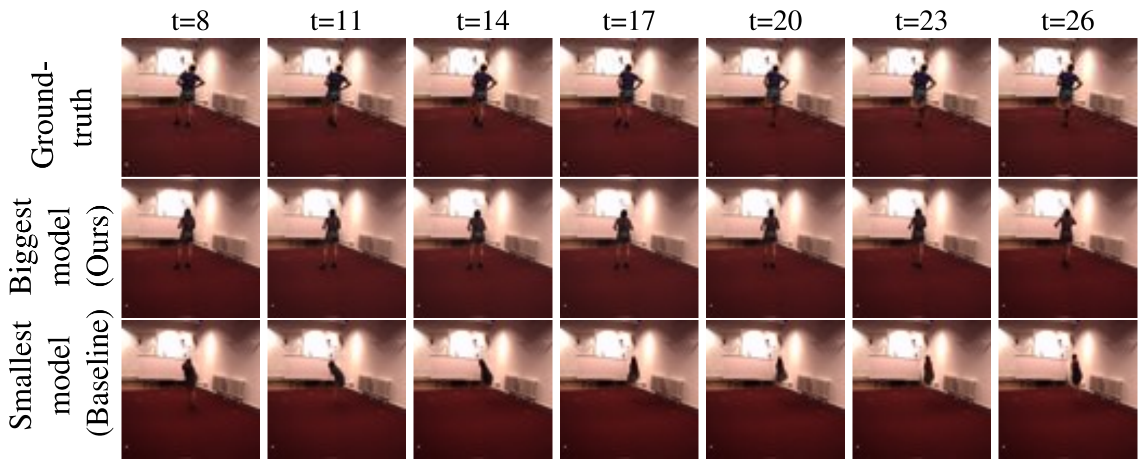


Figure 5.4: Human 3.6M qualitative evaluation. Our highest capacity model (middle) produces better modeling of the human dynamics. The baseline model (bottom) is able to keep the human dynamics to some degree but in often cases the human shape is unrecognizable or constantly vanishing and reappearing. For more videos, please visit our website https://cutt.ly/QGuCex.

the ability to sample multiple futures, leading to a higher probability of matching the ground truth future. Secondly, in our human evaluations for SVG', 95.8% of the AMT workers agree that the bigger model has more realistic videos in comparison to the smaller model, and for LSTM, 98.7% of the workers agree that the LSTM largest model is more realistic. Our results, especially the strong agreement from our human evaluations, show that high capacity models are better equipped to handle the complex structured dynamics in human videos.

Frame-wise evaluation. Similar to the previous per-frame evaluation, we select the best performing models in terms of FVD and perform a frame-wise evaluation. In Figure 5.3, we can see that the CNN based model performs poorly against the LSTM and SVG' baselines. The recurrent connections in LSTM and SVG' are necessary to be able to identify the human structure and the action being performed in the input frames. In contrast to Section 5.4.2, there are no action inputs to guide the video prediction which significantly affects the CNN baseline. The LSTM and SVG' networks perform similarly at the beginning of the video while SVG' outperforms LSTM in the last time steps. This is a result of SVG' being able to model multiple futures from which we pick the best future for evaluation as described in Section 5.4.1. We present the full evaluation on all capacities for SVG', LSTM, and CNN in the supplementary material.

Qualitative evaluation. Figure 5.4 shows a comparison between the smallest and largest stochastic models. In the video generated by the smallest model, the shape of the human is not well-defined at all, while the largest model is able to clearly depict the arms and the legs of the human. Moreover, our large model is able to successfully predict the human's movement throughout all of the frames into the future. The predicted motion is close to the ground-truth motion providing evidence that being able to model more factors of variation with larger capacity models can enable accurate motion identification and prediction. For more videos, please visit our website https://cutt.ly/QGuCex.

5.4.4 Car driving

For this dataset, we also perform action-free video prediction. During training time, the models are conditioned on 5 input frames and predict 10 frames into the future. At test time, the models predict 25 frames into the future. This video type is the most difficult to predict since it requires the model to be able to hallucinate unseen parts in the video given the observed parts.

Dynamics-based evaluation. We see very similar results to the previous dataset when measuring the realism of the videos. For both LSTM and SVG', we see a large improvement in FVD when comparing the baseline model to the largest model we were able to train

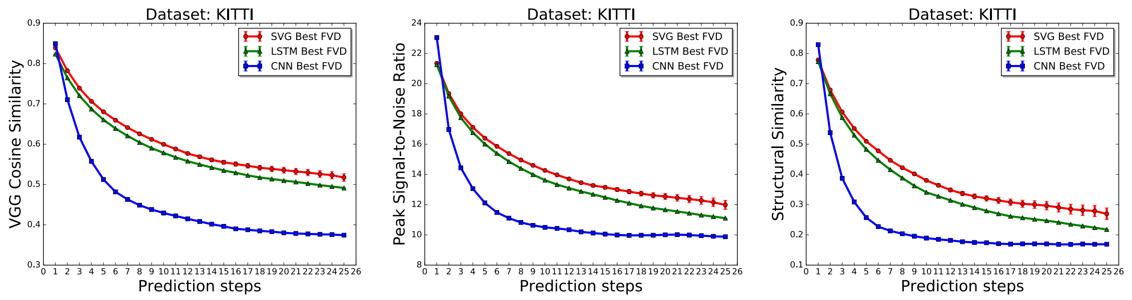


Figure 5.5: KITTI driving per-frame evaluation (higher is better). For model capacity comparisons, please refer to Supplementary C.0.2.3.

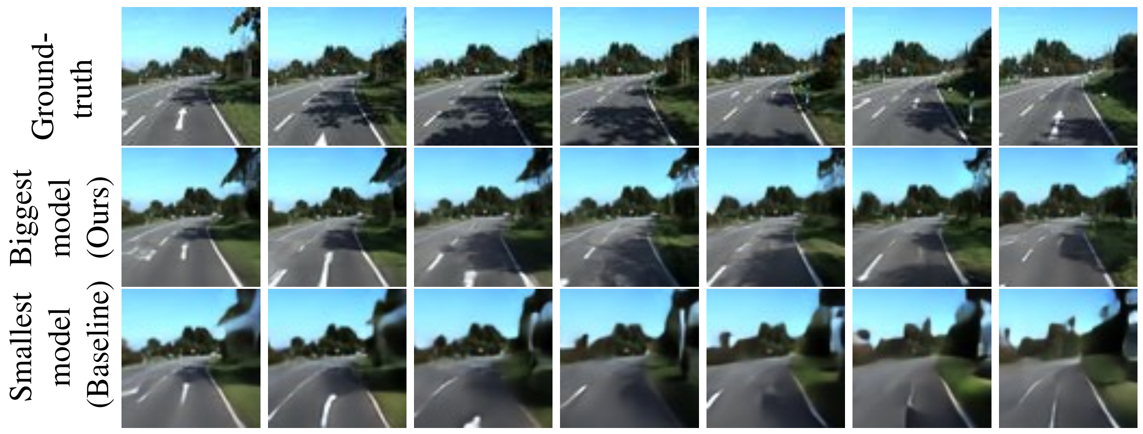


Figure 5.6: KITTI driving qualitative evaluation. Our highest capacity model (middle) is able to maintain the observed dynamics of driving forward and is able to generate unseen street lines and the moving background. The baseline (bottom) loses the street lines and the background becomes blurry. For best viewing and more results, please visit our website `https://cutt.ly/QGuCex`.

(Table 5.1, bottom row). However, we see a similarly poor performance for the CNN architecture as in Section 5.4.3, where capacity does not help. One interesting thing to note is that the largest LSTM model performs better than the largest SVG' model. This is likely related to the architecture design and the data itself. The movements of cars driving is mostly predictable, and so, the deterministic architecture becomes highly competitive as we increase the model capacity (Supplementary C.0.2.3). However, our original premise that increasing model's capacity improves network performance still holds. Finally, for human evaluations, we see in Table 5.2 that the largest capacity SVG' model is preferred by human raters 99.3% of the time (right), and the largest capacity LSTM model (left) is also preferred by human raters 99.3% time (left).

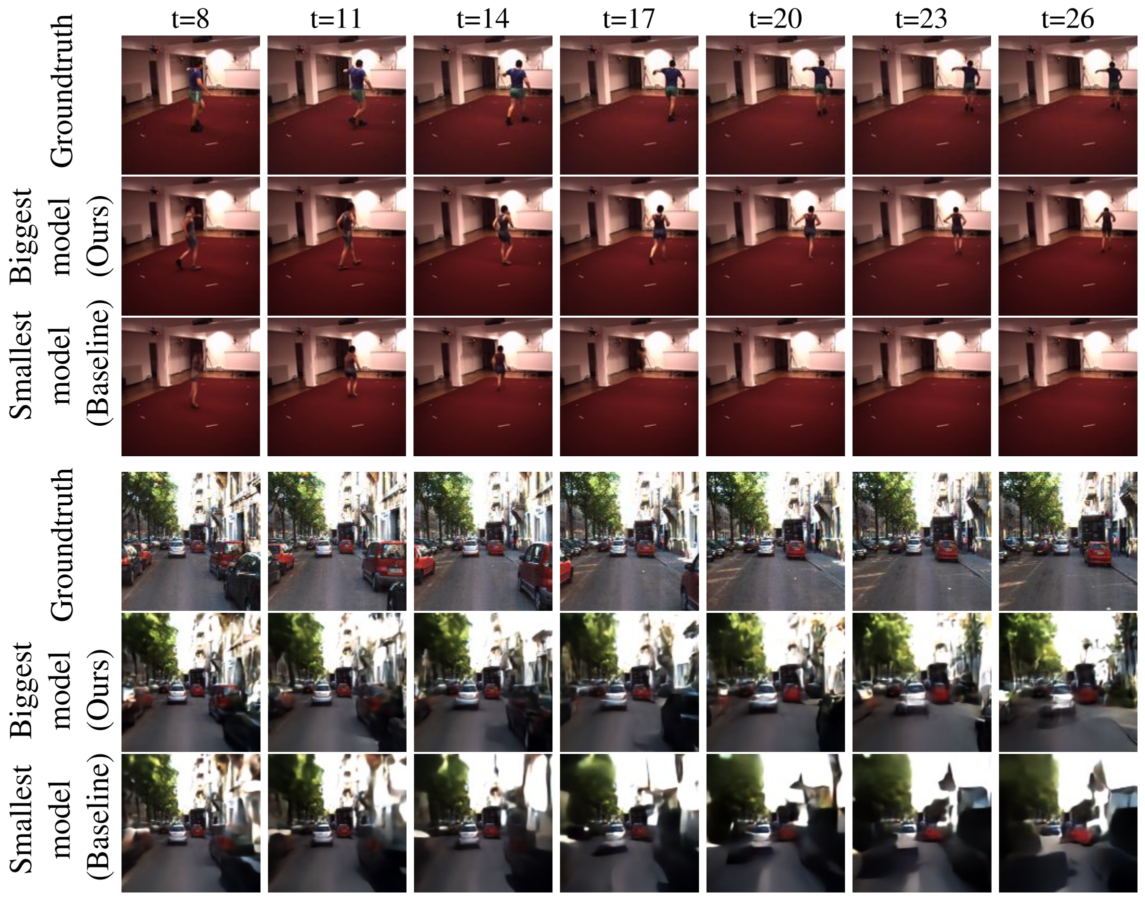


Figure 5.7: Human 3.6M and KITTI driving qualitative evaluation on high resolution videos (frame size of 128x128) with comparison between smallest model and largest model we were able to train (M=3, K=3). For best viewing and more results, please visit our website https://cutt.ly/QGuCex.

Frame-wise evaluation Now, when we evaluate based on frame-wise accuracy, we see similar but not exactly the same behavior as the experiments in Section 5.4.3. The CNN architecture performs poorly as expected, however, LSTM and SVG' perform similarly well.

Qualitative evaluation. In Figure 5.6, we show a comparison between the largest stochastic model and its baseline. The baseline model starts becoming blurry as the predictions move forward in the future, and important features like the lane markings disappear. However, our biggest capacity model makes very sharp predictions that look realistic in comparison to the ground-truth.

5.5 High resolution videos

Finally, we experiment with larger resolution videos. We train SVG' on the Human 3.6M and KITTI driving datasets. These two datasets contain much larger resolution images compared to the Towel pick dataset, enabling us to sub-sample frames to twice the resolution of previous experiments (128x128). We follow the same protocol for the number of input and predicted time steps during training (5 inputs and 10 predictions), and the same protocol for testing (5 inputs and 25 predictions). In contrast to the networks used in the previous experiments, we add three more convolutional layers plus pooling to subsample the input to the same convolutional encoder output resolution as in previous experiments.

In Figure 5.7 we show qualitative results comparing the smallest (baseline) and biggest (Ours) networks. The biggest network we were able to train had a configuration of M=3 and K=3. Higher resolution videos contain more details about the pixel dynamics observed in the frames. This enables the models to have a denser signal, and so, the generated videos become more difficult to distinguish from real videos. Therefore, this result suggests that besides training better and bigger models, we should also more towards larger resolutions. For more examples of videos, please visit our website: https://cutt.ly/QGuCex.

5.6 Conclusion

In conclusion, we provide a full empirical study on the effect of finding minimal inductive bias and increasing model capacity for video generation. We perform a rigorous evaluation with five different metrics to analyze which types of inductive bias are important for generating accurate video dynamics, when combined with large model capacity. Our experiments confirm the importance of recurrent connections and modeling stochasticity in the presence of uncertainty (e.g., videos with unknown action or control). We also find that maximizing the capacity of such models improves the quality of video prediction. We hope our work encourages the field to push along similar directions in the future – i.e., to see how far we can get by finding the right combination of minimal inductive bias and maximal model capacity for achieving high quality video prediction.

CHAPTER VI

Neural Kinematic Networks for Unsupervised Motion Retargeting

We propose a recurrent neural network architecture with a Forward Kinematics layer and cycle consistency based adversarial training objective for unsupervised motion retargeting. Our network captures the high-level properties of an input motion by the forward kinematics layer, and adapts them to a target character with different skeleton bone lengths (e.g., shorter, longer arms etc.). Collecting paired motion training sequences from different characters is expensive. Instead, our network utilizes cycle consistency to learn to solve the Inverse Kinematics problem in an unsupervised manner. Our method works *online*, i.e., it adapts the motion sequence on-the-fly as new frames are received. In our experiments, we use the Mixamo animation data [1] to test our method for a variety of motions and characters and achieve state-of-the-art results. We also demonstrate motion retargeting from monocular human videos to 3D characters using an off-the-shelf 3D pose estimator.

6.1 Introduction

Imitation is an important learning scheme for agents to acquire motor control skills Schaal (1999). It is often formulated as learning from expert demonstrations with access to sample trajectories of state-action pairs Bagnell (2015); Ho and Ermon (2016). However, this first-

[1] https://www.mixamo.com. See details in Section 5.

person imitation assumption may not always hold since 1) the teacher and the learner may have different physical structures, e.g., a human being vs a humanoid robot Bin Hammam et al. (2015); Sermanet et al. (2017) and 2) the learner may only observe the states of the teacher, e.g. joint positions, but not the actions that generate these states Merel et al. (2017). Adapting the motion of the teacher, e.g., a person, to the learner, e.g., a humanoid robot Ayusawa and Yoshida (2017) or an avatar Shon et al. (2006); Mehta et al. (2017), is often referred as motion retargeting in robotics and computer animation. This paper focuses on retargeting motions from a source to any target character with a known but different kinematic structure in terms of bone lengths and proportions. Skeletal differences between the source and target characters create the necessity of disentangling skeleton-independent features of the source motion and automatically adapting them to a target character in one shot, ideally without any post-processing optimization and hand-tuning steps. Furthermore, a faithful solution needs to ensure the retargeted motion to be natural and realistic-looking which has been a long-standing challenge for animation.

Deep neural networks are known to have the ability to learn high-level features in sequential data that humans may not be able to easily identify, and have already achieved remarkable performance in machine translation Johnson et al. (2017) and speech recognition Graves et al. (2013). However, human motions are highly nonlinear and intrinsically constrained by kinematic structures of the skeletons. Thus classic sequence models such as recurrent neural networks (RNNs) may not be directly applicable to motion retargeting.

In this paper, we propose a novel neural network architecture to perform motion retargeting between characters with different skeleton structures (i.e., same topology but different bone length proportions). Our architecture relies on an analytic *Forward Kinematics* layer and two RNNs that work together to (i) encode the input motion data to motion features, and (ii) decode the joint rotations of the target skeleton from the identified features. The forward kinematics layer takes as input the joint rotations and the T-pose of a target skeleton, and renders the resulting motion. This fully differentiable layer forces the network to discover valid joint rotations by enabling to reason about the realism of the resulting motion. We use an adversarial training objective, rooted on the cycle consistency principle Zhu et al. (2017), to learn motion retargeting in an unsupervised way. In particular, the motion retargeted onto a target character should generate the original motion of the source character when retargeted back. Furthermore, the generated motion should be as natural as other known motions of the target character for an adversarially trained discriminator. The decoder RNN is conditioned on the target character, and together with the adverserial training, is able to generate natural motions for unseen characters as well. In our experiments, we show that the proposed method can perform *online* motion retargeting, i.e., adapting the input motion sequence on-the-fly as new frames are received. We also use 3D pose estimates from video sequences, e.g., in Human 3.6M dataset Ionescu et al. (2014b), as input to our network to animate Mixamo 3D characters.

The contributions of our work are summarized below:

- A novel *Neural Kinematic Network* consisting of two RNNs and a forward kinematics layer that automatically discovers the necessary joint rotations (i.e., solution to the *Inverse Kinematics (IK)* problem) for motion retargeting without requiring ground-truth rotations during training.

- A sequence-level adversarial cycle consistency objective function for unsupervised learning for motion retargeting which does not require input/output motion pairs of different skeletons during training.

6.2 Related work

Gleicher Gleicher (1998) first formulated motion retargeting as a spacetime optimization problem with kinematic constraints that is solved for the entire motion sequence. Lee and Shin Lee and Shin (1999) proposed a decomposition approach that first solves the IK problem for each frame to satisfy the constraints and then fits multilevel B-spline curves to achieve smooth results. Tak and Ko Tak and Ko (2005) further added dynamics constraints to perform sequential filtering to render physically plausible motions. Choi and Ko Choi and Ko (1999) proposed an online retargeting method by solving per-frame IK that computes the change in joint angles corresponding to the change in end-effector positions while imposing motion similarity as a secondary task. While the above-mentioned approaches require iterative optimization with hand-designed kinematic constraints for particular motions, our method learns to produce proper and smooth changes of joint angles (solving IK) in one-pass feed-forward inference of RNNs, and is able to generalize to unseen characters and novel motions. The idea of solving approximate IK can be traced back to the early blending-based methods Rose III et al. (2001); Kovar and Gleicher (2004). A target skeleton can be viewed as a new style. Our method can be applied to motion style transfer that has been a popular research area in computer animation Brand and Hertzmann (2000); Hsu et al. (2005); Min et al. (2010); Xia et al. (2015); Yumer and Mitra (2016).

Different machine learning algorithms have been used in modeling human motions. Early works used auto-regressive RBMs Taylor et al. (2007) or Gaussian process dynamic models Wang et al. (2008); Grochow et al. (2004) to learn human motions in small scale. In particular, Grochow et al. Grochow et al. (2004) solves IK by constraining the generated poses to a learned Gaussian process prior. With the surge of deep learning, a variety of neural networks have been used to synthesize human motions Fragkiadaki et al. (2015b); Holden et al. (2016); Jain et al. (2016); Bütepage et al. (2017); Martinez et al. (2017a); Li et al. (2018). These networks are not applicable to motion retargeting as they directly generate the xyz-coordinates of joints and thus require a further post-processing to ensure bone length consistency. Instead, our method predicts quaternions that represent the rotation of each joint with respect to the T-pose without rotation supervision, which admits an end-to-end solution to motion retargeting and also has the potential of synthesizing kinematically plausible motions. Notably, Jain et al. Jain et al. (2016) model human motions with a spatial-temporal graph that considers the skeletal structure but not in an analytic form.

Our work is also related to research efforts on “vision as inverse graphics”. Differentiable rendering layers are incorporated into deep neural networks to disentangle imaging factors of rigid objects, such as 3D shape, camera, normal map, lighting and materials Yan et al. (2016); Rezende et al. (2016); Tulsiani et al. (2017); Liu et al. (2017). Wu et al. Wu et al. (2017) further incorporated a differentiable physics simulator Chang et al. (2017) to disentangle physical properties of multiple rigid objects. Our network disentangles the hierarchical rotations of articulated skeletons through a differentiable forward kinematics layer.

6.3 Background

We first introduce some concepts in robotics and computer animation essential for building our model.

6.3.1 Forward kinematics

Forward kinematics (FK) refers to the process of computing the positions of skeleton joints, also known as *end-effectors*, in 3D space given the joint rotations and initial positions. FK is performed by recursively rotating the joints of an input skeleton tree starting from the root joint and ending in the leaf joints, and is defined by:

$$p^n = p^{parent(n)} + R^n \bar{s}^n,$$

where $p^n \in \mathbb{R}^3$ is the updated 3D position of the n-th joint and $p^{parent(n)} \in \mathbb{R}^3$ is the current position of its parent. $R^n \in \mathbb{SO}(3)$ is the rotation of the n-th joint with respect to its parent. $\bar{s}^n \in \mathbb{R}^3$ is the 3D offset of the n-th joint relative to its parent in the input skeleton, and is defined by:

$$\bar{s}^n = \bar{p}^n - \bar{p}^{parent(n)},$$

Note that $\bar{p}^n$ and $\bar{p}^{parent(n)}$ refer to joint positions in the input *T-pose skeleton* as depicted in Figure 6.1.

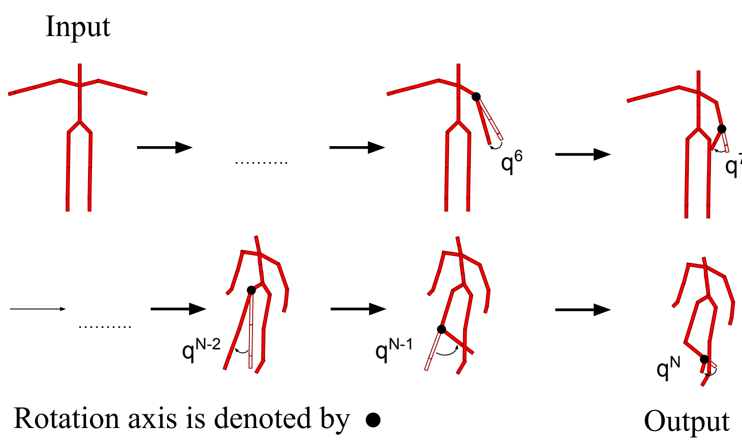


Figure 6.1: Forward kinematics from T-pose skeleton. Starting from the input skeleton, the forward kinematics layer rotates bones to achieve the desired output configuration.

6.3.2 Inverse kinematics

While FK refers to computing the 3D joint locations by recursively applying joint rotations, inverse kinematics (IK) is the reverse process of computing joint rotations $R^{1:N}$ that ensure specific joints are placed at the desired target locations $p^{1:N}$ starting from initial positions $p_0^{1:N}$. Thus, IK is defined by:

$$R^{1:N} = \text{IK}(p^{1:N}, p_0^{1:N}).$$

IK is inherently an ill-posed problem. Target configuration of joint locations can be fulfilled by multiple joint rotations or no joint rotations. Classic IK solutions often resort to iterative optimization by calculating the inverse Jacobian of the highly nonlinear FK function numerically or analytically.

6.4 Method

In this section, we present our proposed method for unsupervised motion retargeting. There are two main components: (i) the neural kinematic network architecture for skeleton conditioned motion synthesis, and (ii) the adversarial cycle consistency training for unsupervised motion retargeting. We next describe these components in detail.

6.4.1 Neural kinematic networks

Our neural kinematic networks for motion synthesis component is built to strictly manipulate a target skeleton, which we refer as *condition skeleton*, into performing a given motion sequence performed by another source skeleton through a *Forward Kinematics* layer.

In our setup, the input motion data $x_{1:T}$ is decomposed into $p_{1:T}$ and $v_{1:T}$, where for each time t, $p_t \in \mathbb{R}^{3N}$ represents the local *xyz*-configuration of the skeleton's pose with respect to its root joint (i.e., hip joint), and $v_t \in \mathbb{R}^4$ represents the global motion of the skeleton's root joint (i.e., *x*,*y*,*z*-velocities and rotation with respect to the axis perpendicular to the ground). Given the condition skeleton, the motion synthesis module outputs the rotations, R_t^n, that are then applied to each joint n at time t, as well as the global motion parameters.

6.4.1.1 Forward kinematics layer

At the core of our neural kinematic networks for motion synthesis component lies the *Forward Kinematics layer* (Figure 6.1) which is designed to take in 3D rotations for each joint n at time t parameterized by *unit quaternions* $q_t^n \in \mathbb{R}^4$, and apply them to a skeleton bone configuration $\bar{s}^n$. A *quaternion* extends a *complex number* in the form $r + x\mathbb{i} + y\mathbb{j} + z\mathbb{k}$ and is used to rotate objects in 3 dimensional space, where r, x, y, and z are real numbers and $\mathbb{i}$, $\mathbb{j}$, $\mathbb{k}$ are quaternion units. The rotation matrix corresponding to an input quaternion is calculated as follows:

$$R_t^n = \begin{pmatrix} 1-2({q_{tj}^n}^2+{q_{tk}^n}^2) & 2(q_{ti}^n q_{tj}^n + q_{tk}^n q_{tr}^n) & 2(q_{ti}^n q_{tk}^n - q_{tj}^n q_{tr}^n) \\ 2(q_{ti}^n q_{tj}^n - q_{tk}^n q_{tr}^n) & 1-2({q_{ti}^n}^2+{q_{tk}^n}^2) & 2(q_{tj}^n q_{tk}^n + q_{ti}^n q_{tr}^n) \\ 2(q_{ti}^n q_{tk}^n + q_{tj}^n q_{tr}^n) & 2(q_{tj}^n q_{tk}^n - q_{ti}^n q_{tr}^n) & 1-2({q_{ti}^n}^2+{q_{tj}^n}^2) \end{pmatrix} \tag{6.1}$$

Given the rotation matrices $R_t^n \in \mathbb{SO}(3)$ for each joint, the FK layer updates the joint positions of the condition skeleton by applying these rotations in a recursive manner as described in Section 6.3.1 and shown in Figure 6.1,

$$p_t^{1:N} = \text{FK}(q_t^{1:N}, \bar{s}).$$

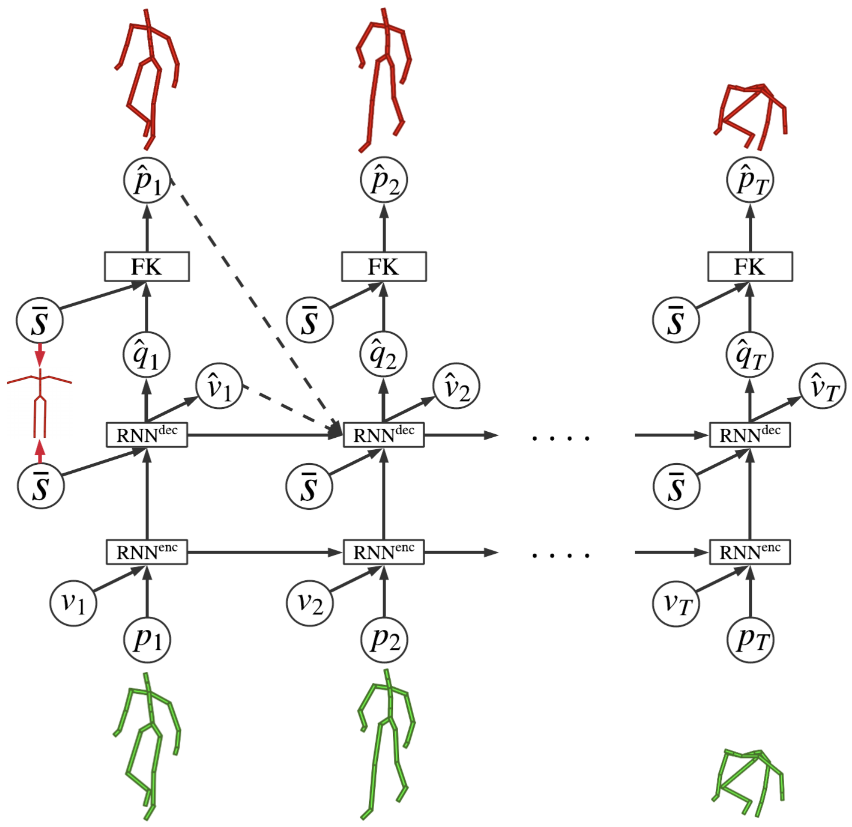


Figure 6.2: Neural kinematic networks for motion synthesis.

The FK layer serves as a tool for mapping the joint rotations to actual joint locations and thus helps our network to focus on learning skeleton independent motion features, i.e., joint rotations.

6.4.1.2 Online motion synthesis

Our proposed neural kinematic networks architecture for online motion synthesis is shown in Figure 6.2. Taking advantage of the temporal coherency in motion sequences, we synthesize the current motion step at time t by conditioning on previous steps through an RNN hidden representation.

The current step in the input motion is encoded by:

$$h_t^{\text{enc}} = \text{RNN}^{\text{enc}}(x_t, h_{t-1}^{\text{enc}}; W^{enc}), \tag{6.2}$$

where $\text{RNN}^{\text{enc}}(.,.)$ is an encoder RNN, h_t^{enc} is the encoding of the input motion up to time t, and $x_t = [p_t, v_t]$ is the current input. The encoded feature is then fed to a decoder RNN to perform skeleton conditioned motion synthesis by:

$$\begin{aligned}
h_t^{\text{dec}} &= \text{RNN}^{\text{dec}}(\hat{x}_{t-1}, h_t^{\text{enc}}, \bar{s}, h_{t-1}^{\text{dec}}; W^{dec}), && (6.3)\\
\hat{q}_t &= \frac{W^{pT} h_t^{\text{dec}}}{\|W^{pT} h_t^{\text{dec}}\|}, && (6.4)\\
\hat{p}_t &= \text{FK}(\hat{q}_t, \bar{s}), && (6.5)\\
\hat{v}_t &= W^{vT} h_t^{\text{dec}}, && (6.6)\\
\hat{x}_t &= [\hat{p}_t, \hat{v}_t]\,. && (6.7)
\end{aligned}$$

where h_t^{dec} is the hidden representation of decoder RNN, $\hat{x}_t$ is the synthesized motion at time step t for the condition skeleton $\bar{s}$. The unit vector $\hat{q}_t \in \mathbb{R}^{4N}$ denotes the rotations — which can be interpreted as actions — to be applied to the condition skeleton through the FK layer. The outputs $\hat{p}_t$ and $\hat{v}_t$ are the estimated local and global motion of the condition skeleton. Finally, W^{enc}, W^{dec}, $W^v \in \mathbb{R}^{d\times 4}$ and $W^p \in \mathbb{R}^{d\times 4N}$ are learnable parameters.

When the condition skeleton is different from the skeleton where the input motion lives, the decoder is meant to generate the rotations of a new character to achieve motion retargeting. Please note that in the rest of the paper, we use superscripts A or B to refer to the identity of the skeleton we are retargeting motion from and into.

6.4.2 Adversarial cycle training for unsupervised motion retargeting

In Section 6.4.1, we describe a method for skeleton conditioned motion synthesis based on a forward kinematics layer embedded within the network architecture. However, training such a network for motion retargeting is challenging as it is very expensive to collect paired motion data x_t^A and x_t^B where the same motion is performed by two different skeletons. Note that collecting such data requires using iterative optimization based IK methods in addition to human hand-tuning of the retargeted motion.

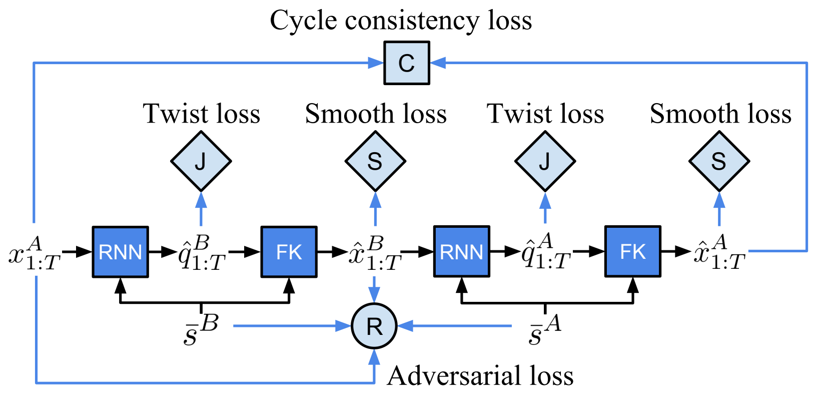


Figure 6.3: Adversarial cycle consistency framework.

We propose a training paradigm based on the cycle consistency principle Zhou et al. (2016) and adversarial training Goodfellow et al. (2014a) for unsupervised motion retargeting (Figure 6.3). Let f be our neural kinematic network, and let the superscripts define skeleton identity. Given an input motion sequence from skeleton A, we first retarget the input motion to skeleton B and back to A as follows:

$$\hat{x}^B_{1:T} = f(x^A_{1:T}, \bar{s}^B), \tag{6.8}$$

$$\hat{x}^A_{1:T} = f(\hat{x}^B_{1:T}, \bar{s}^A), \tag{6.9}$$

where $\hat{x}^B_{1:T}$ and $\hat{x}^A_{1:T}$ are synthesized motions for skeletons B and A, respectively. Therefore, we define four loss terms: adversarial loss on $\hat{x}^B_{1:T}$, cycle consistency loss on $\hat{x}^A_{1:T}$, twist loss on rotations $\hat{q}^A_{1:T}$ and $\hat{q}^B_{1:T}$, and smoothing loss on $\hat{v}^A_t$ and $\hat{v}^B_t$, so our full training objective is defined by:

$$\begin{aligned} \min_f \max_d \quad & C(\hat{x}^A_{1:T}, x^A_{1:T}) + R(\hat{x}^B_{1:T}, x^A_{1:T}) + \\ & \lambda\, J(\hat{q}^B_{1:T}, \hat{q}^A_{1:T}) + \omega\, S(\hat{v}^B_{1:T}, \hat{v}^A_{1:T}), \end{aligned} \tag{6.10}$$

where C is the cycle consistency loss, R the adversarial loss, J the joint twist loss, and S the velocity smoothing loss.

Adversarial loss. The input motion $x^A_{1:T} = \left[p^A_{1:T}, v^A_{1:T}\right]$, the synthesized motion $\hat{x}^B_{1:T} = \left[\hat{p}^B_{1:T}, \hat{v}^B_{1:T}\right]$, and their respective skeleton are fed to a discriminator network g that computes a *realism score* for *real* and *fake* motion sequences:

$$r^A = g(p^A_{2:T} - p^A_{1:T-1}, v^A_{1:T-1}, \bar{s}^A), \tag{6.11}$$

$$r^B = g(\hat{p}^B_{2:T} - \hat{p}^B_{1:T-1}, \hat{v}^B_{1:T-1}, \bar{s}^B), \tag{6.12}$$

where r^A is the output of the discriminator given real data, and r^B is the output of the discriminator given the fake data (i.e., the motion retargeted by our network into skeleton B). The inputs to the discriminator $p^A_{2:T} - p^A_{1:T-1}$ and $p^B_{2:T} - p^B_{1:T-1}$ are the local motion difference between two adjacent time steps, and $\bar{s}^A$ and $\bar{s}^B$ denote the input and target skeletons A and B, respectively. During training, we randomly sample $\bar{s}^B$ from all the available skeletons, thus, it is possible for skeleton B to be the same as skeleton A. In case skeleton B is the same as skeleton A, $\hat{x}^B_{1:T} = \hat{x}^A_{1:T}$, we switch between adversarial and square loss as follows:

$$R(\hat{x}^B_{1:T}, x^A_{1:T}) = \begin{cases} \|\hat{x}^B_{1:T} - x^A_{1:T}\|^2_2, & \text{if } B = A \\ \log r^A + \beta \log(1 - r^B), & \text{otherwise.} \end{cases}, \tag{6.13}$$

When B and A are not the same, we rely on the motion distributions learned by g as a training signal. By observing other motion sequences performed by skeleton B, the discriminator network learns to identify motion behaviors of skeleton B. The generator (encoder and decoder RNNs) uses this as indirect guidance to learn how the motion should be retargeted to B and thus fool the discriminator. When applying the adversarial loss, we use a balancing term β to regulate the strength of the discriminator signal when optimizing f to fool g. We use $\beta = 0.001$ in our experiments.

Cycle consistency loss. The cycle consistency loss C optimizes the following objective:

$$C(\hat{x}^A_{1:T}, x^A_{1:T}) = \|x^A_{1:T} - \hat{x}^A_{1:T}\|^2_2. \tag{6.14}$$

Equation 6.14 encourages f to be able to take its own retargeted motion and map it back to the original motion source effectively achieving cycle consistency.

Twist loss. By optimizing the first two terms in Equation 6.10, our network discovers the necessary rotations to move the input skeleton end-effectors to the required positions for motion retargeting. However, this does not prevent potential excessive bone twisting since *xyz*-coordinates can be perfectly predicted regardless of how many times we rotate a bone around its own axis. Thus, the third term in our objective constrains the bone rotations around its own axis.

$$\begin{aligned} J(\hat{q}^B_{1:T}, \hat{q}^A_{1:T}) =& \| \max(\mathbf{0}, |euler_y(\hat{q}^B_{1:T})| - \alpha) \|_2^2 + \\ & \| \max(\mathbf{0}, |euler_y(\hat{q}^A_{1:T})| - \alpha) \|_2^2, \end{aligned} \tag{6.15}$$

where $euler_y(.)$ converts the quaternion outputs of our network into rotation angles around the standard *xyz*-axes and the subscript y means to select the rotation angle around the plane parallel to the bone (i.e. y-axis). Therefore, any bone rotation exceeding α degrees in either negative or positive direction is penalized in our objective function. We use $\alpha = 100°$, and $\lambda = 10$ in our experiments.

Smoothing loss. Finally, the first two terms in our objective function treat global motion at each time step independently. However, global motion in consecutive timesteps are highly dependent on each other, that is, global motion in the next timestep should change only slightly with respect to the previous global motion. We constraint the global motion by:

$$\begin{aligned} S(\hat{v}^B_{1:T}, \hat{v}^A_{1:T}) =& \| \hat{v}^B_{2:T} - \hat{v}^B_{1:T-1} \|_2^2 + \\ & \| \hat{v}^A_{2:T} - \hat{v}^A_{1:T-1} \|_2^2, \end{aligned} \tag{6.16}$$

We use $\omega = 0.01$ in our experiments.

6.5 Experiments

Dataset. We evaluate our method on the Mixamo dataset Adobe's Mixamo (2017) which contains approximately 2400 unique motion sequences for 71 characters (i.e., skeletons). For training, we collected non-overlapping motion sequences for 7 characters (AJ, Big Vegas, Goblin Shareyko, Kaya, Malcolm, Peasant Man, and Warrok Kurniawan) which in total results in 1646 training sequences at 30 frames per second.

For testing, we collected motion sequences for 6 characters (Malcolm, Mutant, Warrok Kurniawan, Sporty Granny, Claire, and Liam) and perform retargeting in four scenarios:

- Input motion is seen during training, and the target character is also seen during training but the target motion sequence is not.
- Input motion is seen during training but the target character is never seen during training.
- Input motion is not seen during training but the target character is seen during training.
- Neither the input motion nor the target character are seen during training.

Note that we also collected the ground truth retargeted motions of testing sequences for quantitative evaluation purposes only. While we discuss our main findings below, detailed results and analysis of each scenario and character can be found in the appendix as well as details of how to acquire the exact training and testing data.

Data preprocessing. Each motion sequence is pre-processed by separating into local and global motion, similar to Holden et al. (2016). For local motion, we remove the global displacement (i.e., the motion of the root joint), and rotation around the axis vertical to the ground. Global motion consists of the velocity of the root in the x, y, and z directions, and an additional value representing the rotation around the axis perpendicular to the ground. For training, and testing we use the following 22 joints: Root, Spine, Spine1, Spine2, Neck, Head, LeftUpLeg, LeftLeg, LeftFoot, LeftToeBase, RightUpLeg, RightLeg, RightFoot, RightToeBase, LeftShoulder, LeftArm, LeftForeArm, LeftHand, RightShoulder, RightArm, RightForeArm, and RightHand.

Baseline methods. While there have been several optimization based approaches for the IK problem, most of these expect the user to provide motion specific constraints or goals. Since this is not feasible to do at a large scale, we instead show comparisons to learning based baseline methods that aim to identify such constraints automatically. The first baseline is an RNN architecture without the FK layer that directly outputs *xyz*-coordinates for the local motion, and the global motion output is the same as ours. Second, we use an MLP architecture that lacks recurrent connections, and directly outputs the *xyz*-coordinates for the local motion, and the same global motion output as our method. We also train both baselines with our adversarial cycle consistency objective. Finally, we include another baseline that directly copies the per-joint rotation and the global motion of the input motion into the target skeleton.

Training and evaluation. We train our method and baselines by randomly sampling 2-second motion clips (60 frames) from the training sequences, and testing on motion clips of 4 seconds (120 frames) from the test sequences. We initialized the quaternion outputs of the decoder RNN to be close to the identity rotation (i.e., close to zero rotation). For training the discriminator network, we sample random motion sequences being performed by the same skeleton into which the motion synthesis network is retargeting motion. Details of the network architecture and hyperparameters can be found in the appendix. We perform two types of evaluations: 1) We evaluate the overall quality of the motion retargeting using a target character normalized Mean Square Error (MSE) on the estimated joint locations through time (i.e., *xyz*-coordinates after combining local and global motion together). 2) We compare end-effector locations through time against the ground-truth. 3) We show qualitative results by rendering the animated 3D characters using the outputs of our network.

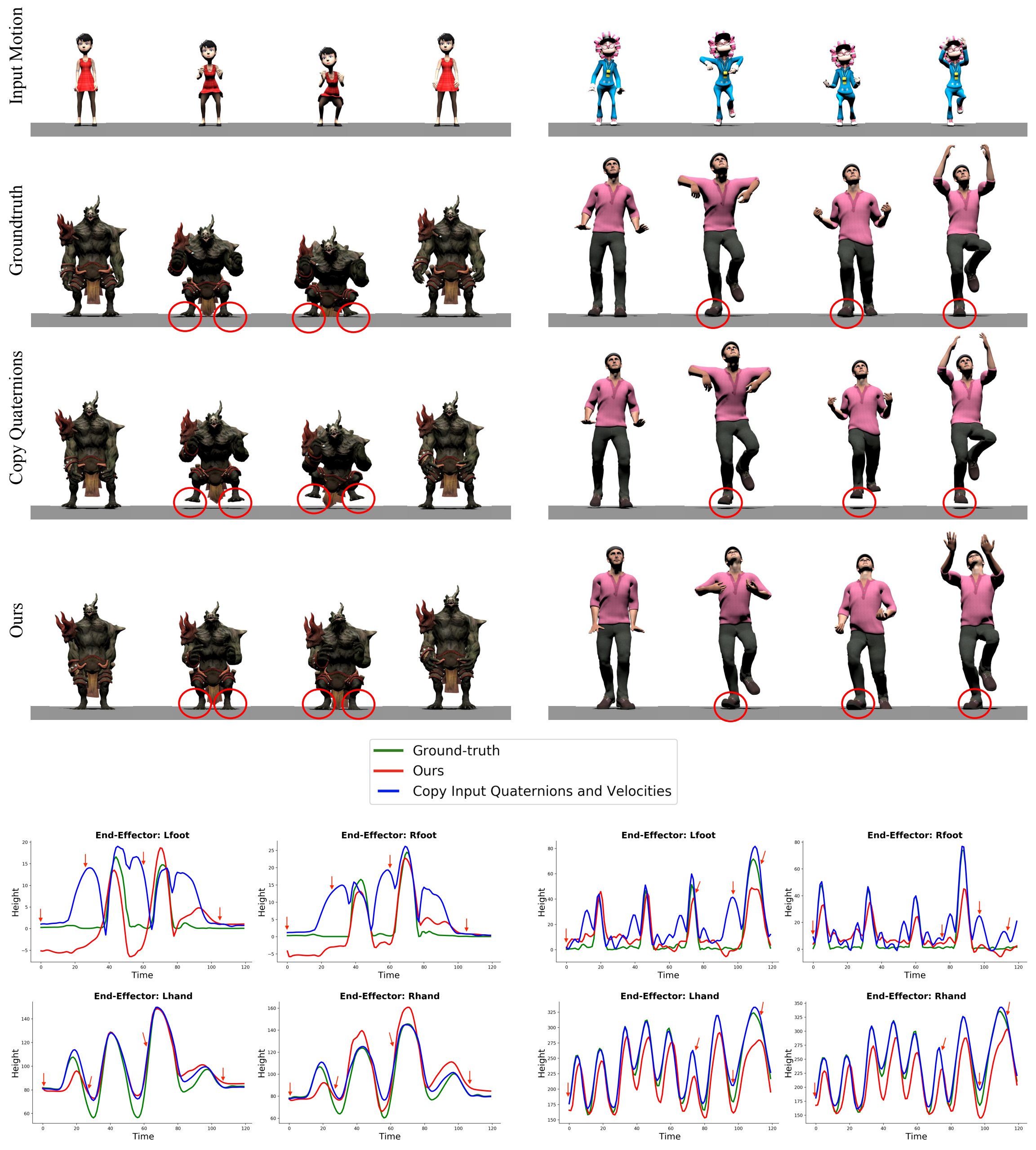


Figure 6.4: Qualitative evaluation. We present a motion retargeting example of our method against the best baseline. Motion is retargeted from character Claire into Warrok Kurniawan (left) and Sporty Granny to Malcolm (right). Plots illustrating the left/right feet and hand end-effectors' height comparing against the groundtruth are shown at the bottom. Arrows in the plots determine the time steps of the shown animation frames. Please visit goo.gl/mDTvem for animated videos.

Method	MSE
Ours: Autoencoder Objective	10.25
Ours: Cycle Consistency Objective	8.51
Ours: Adversarial Cycle Consistency Objective	**7.10**
Baseline: Conditional RNN	13.65
Baseline: Conditional RNN + Adv. Cycle Consistency	26.93
Baseline: Conditional MLP	17.02
Baseline: Conditional MLP + Adv. Cycle Consistency	16.96
Baseline: Copy input quaternions and velocities	9.00

Table 6.1: Quantitative evaluation of online motion retargeting using mean square error (MSE).

6.5.1 Online Motion retargeting From Character

In this section we evaluate our method on the task of online motion retargeting, i.e., retargeting motion from one character to a target character as new motion frames are received. We present an ablation study to demonstrate the benefits of the different components of our method, and also compare against the previously described baselines. In Table 6.1, we report the average MSE of the retargeted motion when our network is trained with different objectives: 1) Our skeleton conditioned motion synthesis network (Section 6.4.1) trained with the autoencoder objective (i.e., input reconstruction) and the bone twisting constraint only. 2) Our network trained with the cycle consistency objective without adversarial training. Specifically, the "otherwise" branch in Equation 6.13, returns 0. 3) Our network trained with our full adversarial cycle consistency objective function which requires examples of motions performed by skeleton B but not paired with any motions used as inputs during training.

As it can be seen in Table 6.1, simply using the proposed FK layer within RNNs and training with an autoencoder objective (Ours: Autoencoder Objective), outperforms all standard neural network based baselines. One explanation is that it is highly probable for the baselines to ignore the bone lengths of the target skeleton, and learn a motion representation that is dependent on the input skeleton. The inability to disentangle motion properties from the input skeleton is more evident after training with our adversarial cycle consistency objective which still results in poor performance. The inputs to the discriminator network are velocities, that is, local motion difference between adjacent time steps and global motion. While this input contains information about the shift in joint locations through time, it does not capture any information about the spatial structure. As a result, optimizing the baselines to fool the discriminator network, does not impose bone length constraints. Furthermore, encouraging velocities to be similar to the real data causes further bone length degradation (i.e., excessive stretching or shrinking) in absence of such constraints. On the other hand, our architecture is designed to learn a skeleton invariant motion representation that can be directly transferred to the target skeleton through the FK layer.

The performance of our method improves when training our motion synthesis network with the proposed objectives for cycle consistency and adversarial cycle consistency. While training with the autoencoder objective results in reasonable performance, often the network tries to match end-effector locations but does not fully capture the properties of the input motion. For example, when an input motion of a small character raising its hands is retargeted to a very tall character, the tall character is likely not able to raise its hands but only point in the same direction as the input motion. Our network improves when trained with the cycle consistency objective alone. In the example of motion retargeting from a small to a tall character, cycle consistency loss prevents the tall character from directly matching end-effector positions of the small character as retargeting back to the small character would have resulted in stretching the limbs in the small character. The cycle consistency encourages the network to better learn the high level features of the input motion.

Finally, our method performs the best when our objective imposes both cycle consistency and realism via the full adversarial cycle consistency objective. The adversarial training helps the network to produce motions that cannot be distinguished from realistic motions of the target character.

The baseline "Copy input quaternions and velocities" works better than the neural network baselines due to the following reasons: 1) Copying per-joint rotations of the input and performing forward kinematics already respects the target skeleton bone lengths, and 2)

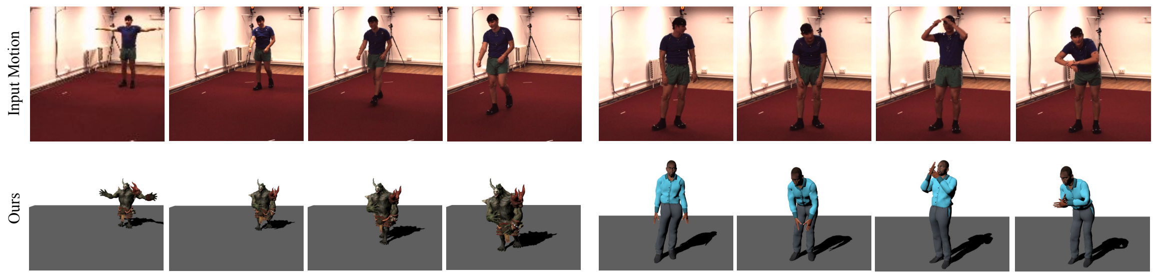


Figure 6.5: Qualitative evaluation on human videos. Motion is retargeted from estimated 3D pose from the Human 3.6M dataset into Mixamo 3D characters using the estimated 3D pose from Martinez et al. (2017b). Please visit `goo.gl/mDTvem` for animated videos.

copying the velocities (i.e., global motion) avoids drifting that prediction models may suffer from. However, when retargeting motions between characters with significant skeleton difference, this baseline is prone to artifacts such as foot floating (see Figure 6.4). This baseline is also not scalable to cases where different skeleton limits or topological structures are considered.

In Figure 6.4, we show qualitative results where we animate target characters using the output of our network using Blender Blender Online Community (2017), a character animation software. For all the joints that are not modeled by our network (e.g., the fingers), we simply directly copy the joint rotations from the input motion if the corresponding joint names match in the input and the target skeleton, otherwise we leave them fixed.

6.5.2 Online Motion retargeting from Human Video

In this section we present motion retargeting from human video input into characters using the model trained from the Mixamo data only. We use the Human 3.6M videos as input, the algorithm from Martinez et al. (2017b) to estimate the 3D pose of each frame, and the ground truth 3D skeleton root displacement (3D pose estimation algorithms usually assume the person is centered). The videos are subsampled to 25 FPS, and the estimated 3D poses are processed similar to our previous experiment. The algorithm in Martinez et al. (2017b) only outputs 17 joints compared to the 22 joints needed by our network. Therefore, we manually map the 17 joints to 22 by duplicating the following joint positions in Human

3.6M to corresponding Mixamo joints: Spine into Spine and Spine1, LeftShoulder into LeftShoulder and LeftArm, RightShoulder into RightShoulder and RightArm , LeftFoot into LeftFoot and LeftToeBase, RightFoot into RightFoot and RightToeBase. Note that this mapping will create bones of zero length during test time. Thus, our network essentially only sees 17 joints but uses 22 joints as input. During visualization, we do not rotate joints that are not predicted by our network (i.e., fingers). As shown in Figure 6.5, our network is able to generalize to never-seen skeletons and motions estimated from monocular human videos. More video results and analyses are included in appendix.

6.6 Conclusion and Future Work

We have presented a neural kinematic network with an adversarial cycle consistency training objective for motion retargeting. Our network only observes a sequence of xyz-coordinates of joints from existing animations, motion capture or 3D pose estimates of monocular human videos, and transfers the motion to a target humanoid character without risking skeleton deformations that occur in the baselines. The success of our method attributes to the following factors: 1) The proposed *Forward Kinematics layer* helps to discover joint rotations of target skeleton that are independent of the input skeleton. 2) The cycle consistency of the retargeting objective prevents regressing to the end-effector positions of the input motion. 3) The adversarial objective helps the network to produce realistic motions. 4) The bone twist loss constrains the solution space of *Inverse Kinematics* and prevents bone twisting in the retargeted motion.

Our current method has limitations. First, we perform retargeting on a fixed number of joints. Handling a variable number of joints is challenging as the retargeting algorithm is expected to automatically select end-effectors of interest when transferring motions. Second, we assume the environment in which the target character is being animated lacks physical constraints such as gravity. Future work will include equipping the network with physics simulators to generate more natural and physically plausible movements of the target characters with different muscle/bone mass distributions. Third, the input to our method still requires 3D information (xyz-coordinates of joints). Future work will also include training our network end-to-end by using monocular videos as input. That may require the algorithm to learn view-invariant features.

CHAPTER VII

Discussion and Future Work

In this thesis, I have proposed algorithms for modeling structured dynamics in 2D and 3D. I start with separating the motion and contents from the input sequence into separate streams. By having a separate stream handling the motion, I encourage the neural network identify useful information from each of the groups to predict future dynamics more easily. Next, I model the motion stream as a more compact human made representation that can be verified and directly used for generating future frame dynamics. I model the dynamics as landmarks that define the shapes of objects to be predicted into the future, and use a separate image generation module to generate future pixel of the objects. Subsequently, we proposed a method that learns a general representation of object structures rather than landmarks which also achieves long-term frame prediction. I then investigate maximizing the capacity of neural network based architectures for dynamics models while minimizing inductive bias for high quality future frame prediction. I shine light of the importance of network capacity for better generative models of video with multiple different challenges going from object interactions; to highly structured objects; to moving background. Finally, I move on to modeling dynamics in 3D by using a kinematic structure of humanoid shape (the method can generalize to any hierarchical kinematic shape). The proposed method have benefits and potential future work in the following application areas:

Model-based reinforcement learning:

The model-based reinforcement learning literature has begun to take notice and advantage of the use of prediction models from pixels. Having models that can separate environment dynamics from the objects in the environment can enable agents to more accurately reason about previous and future state dynamics, and also current state configuration. The models in Chapter II could be easily used for current model-based reinforcement learning approaches. However, the model in Chapter III may not be straight forwardly used and will need to discover landmarks from videos in an unsupervised way. This is addressed in a current collaboration that is under review.

Video/Motion synthesis:

Methods that can accurately model dynamics can also be used as a motion synthesis tool. Particularly, a generative or conditional-generative model can enable creation of new motions by sampling from a distribution. This can enable automatic diverse content generation such as, in the case of humans, walking, dancing, jumping, etc animations that can then be used as condition for video generation. The works presented in Chapter III and VI can be extended with minor changes to make generative models for motion synthesis in both, 2D and 3D domains. The work in Chapter II will require a more significant change to be able to locate regions of interest on which to apply motion, however, it has the most potential for generality in the long run.

Human/object interaction:

An accurate model of structured dynamics such as human dynamics is useful on its own. However, there are usually more than a single human/objects present in real world scenarios. Therefore, an extension that considers possible human interactions during a prediction is necessary. The works in Chapter III and VI can be extended by having multiple recurrent neural networks where each tracks the dynamics of a single human or object in the scene. The recurrent neural networks then communicate with each other such that the independent dynamics are consistent with each other. Works on tracking human trajectory interactions have been recently proposed (Alexandre Alahi and Kratarth Goel and Vignesh Ramanathan and Alexandre Robicquet and Li Fei-Fei and Silvio Savarese, 2016), however, the detailed body structure or general objects present is not considered.

APPENDICES

APPENDIX A

Decomposing Motion and Content for Natural Video Prediction

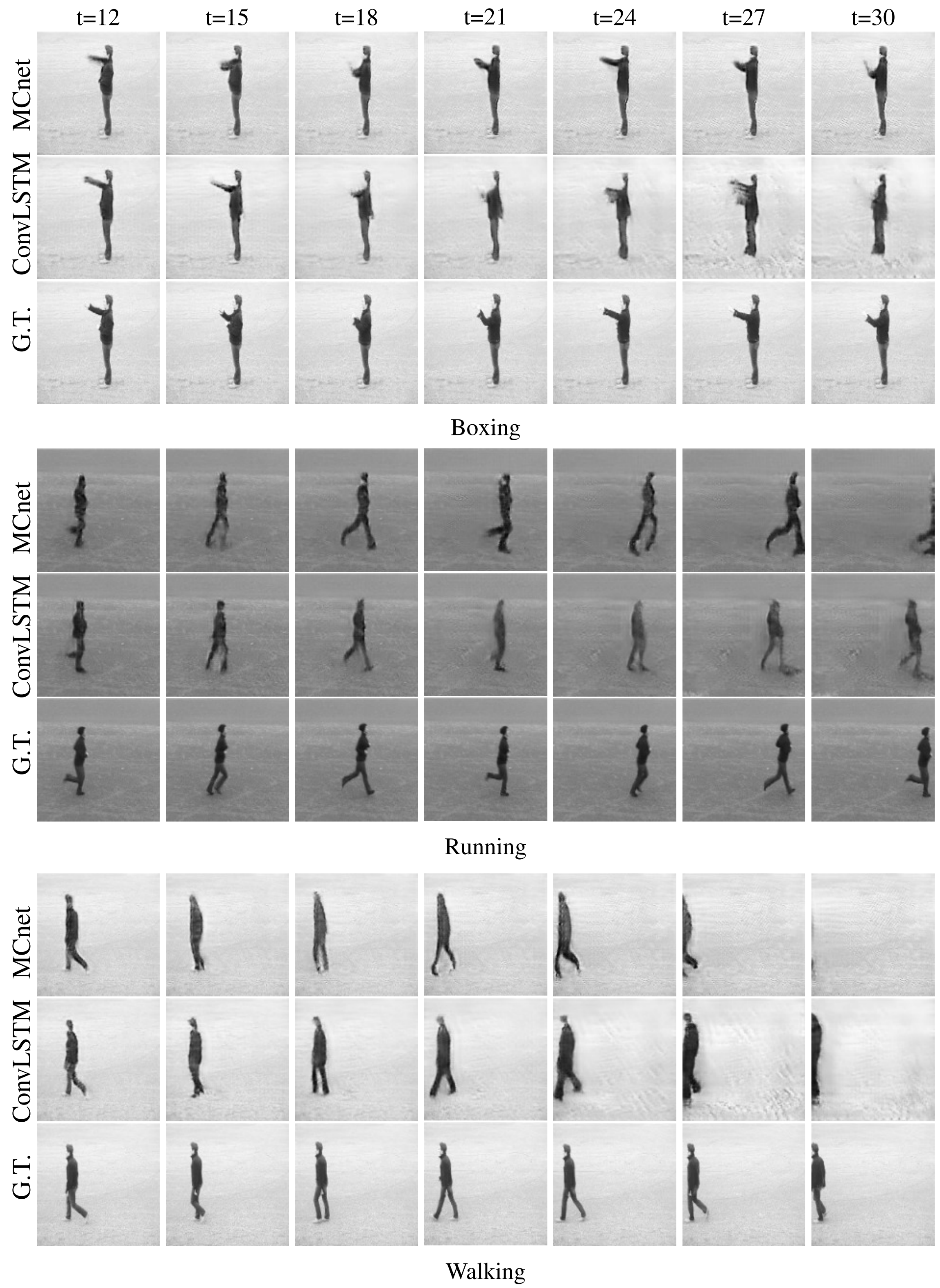
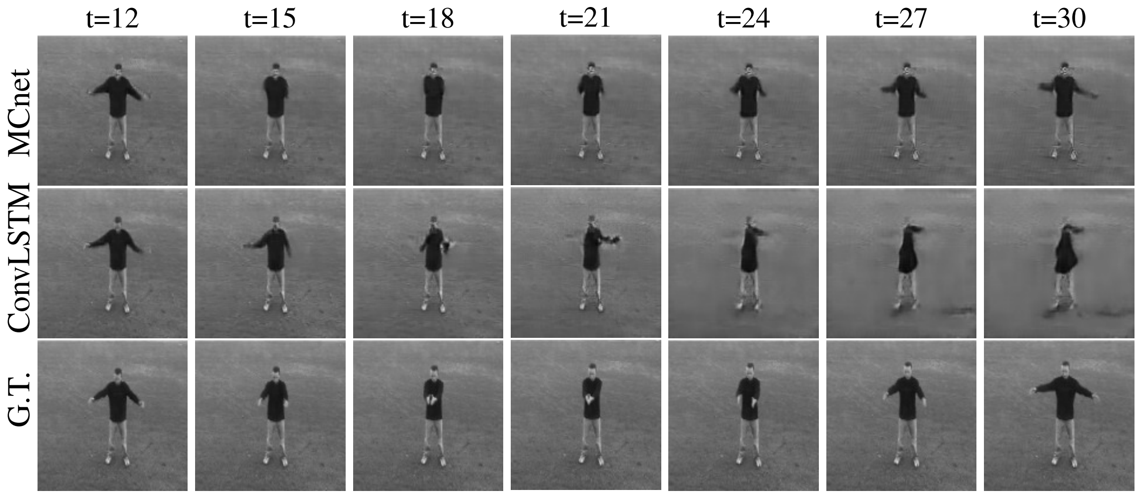
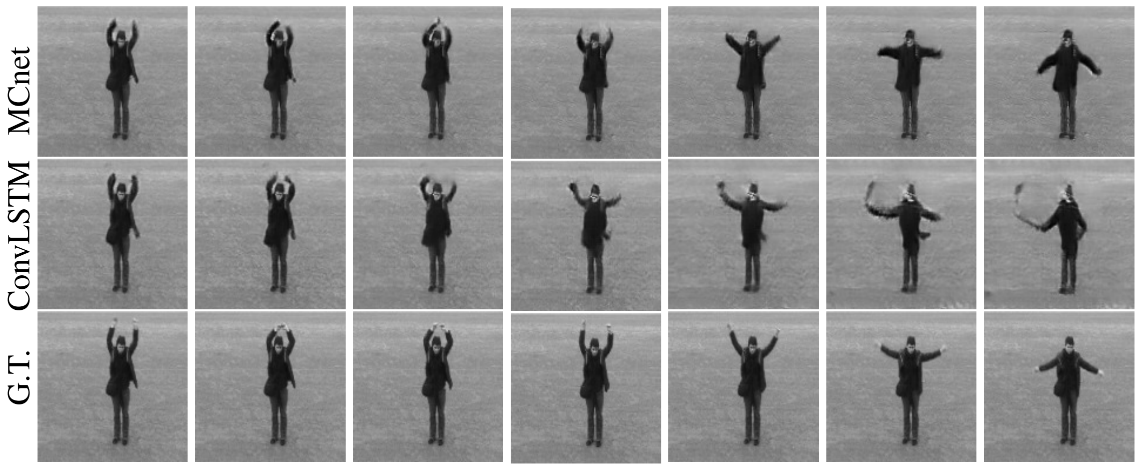


Figure A.1: Qualitative comparisons on KTH testset. We display predictions starting from the 12th frame, for every 3 timesteps. More clear motion prediction can be seen in the project website.



Handclap



Handwaving

Figure A.2: Qualitative comparisons on KTH testset. We display predictions starting from the 12th frame, for every 3 timesteps. More clear motion prediction can be seen in the project website.

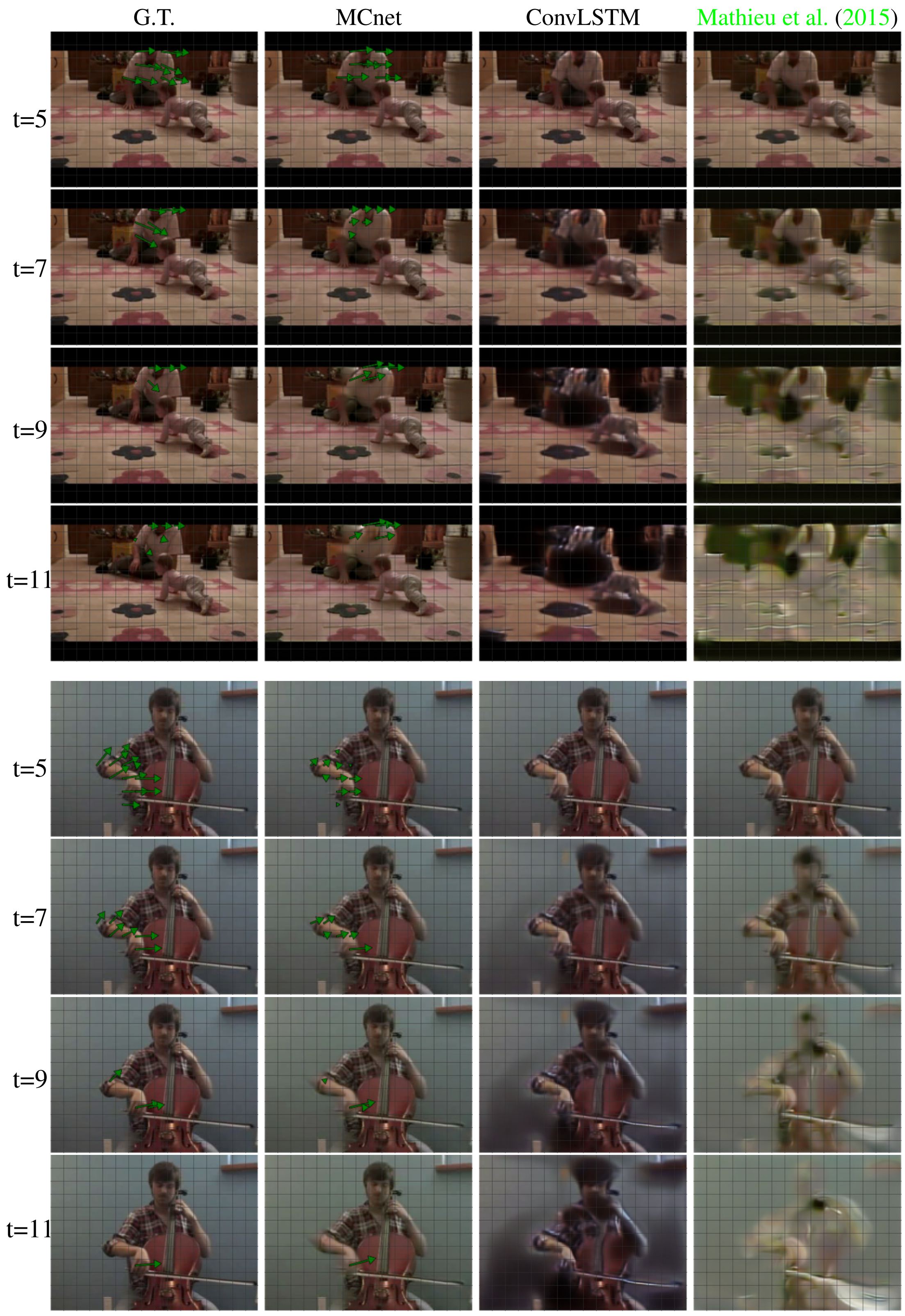


Figure A.3: Qualitative comparisons on UCF-101. We display predictions (in every other frame) starting from the 5th frame. The green arrows denote the top-30 closest optical flow vectors within image patches between MCnet and ground-truth. More clear motion prediction can be seen in the project website.

A.1 Qualitative and quantitative comparison with considerable camera motion and analysis

In this section, we show frame prediction examples in which considerable camera motion occurs. We analyze the effects of camera motion on our best network and the corresponding baselines. First, we analyze qualitative examples on UCF101 (more complicated camera motion) and then on KTH (zoom-in and zoom-out camera effect).

UCF101 Results. As seen in Figure A.4 and Figure A.5, our model handles foreground and camera motion for a few steps. We hypothesize that for the first few steps, motion signals from images are clear. However, as images are predicted, motion signals start to deteriorate due to prediction errors. When a considerable amount of camera motion is present in image sequences, the motion signals are very dense. As predictions evolve into the future, our motion encoder has to handle large motion deterioration due to prediction errors, which cause motion signals to get easily confused and lost quickly.

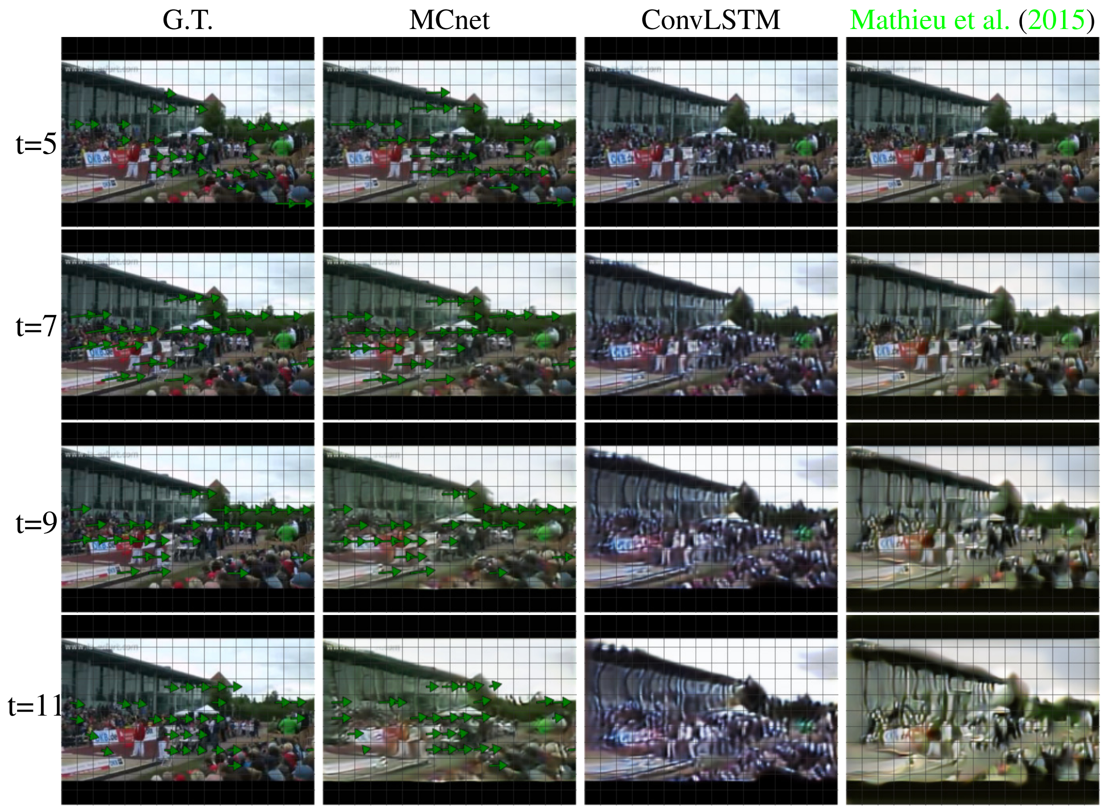


Figure A.4: Qualitative comparisons on UCF-101. We display predictions (in every other frame) starting from the 5th frame. The green arrows denote the top-30 closest optical flow vectors within image patches between MCnet and ground-truth. More clear motion prediction can be seen in the project website.

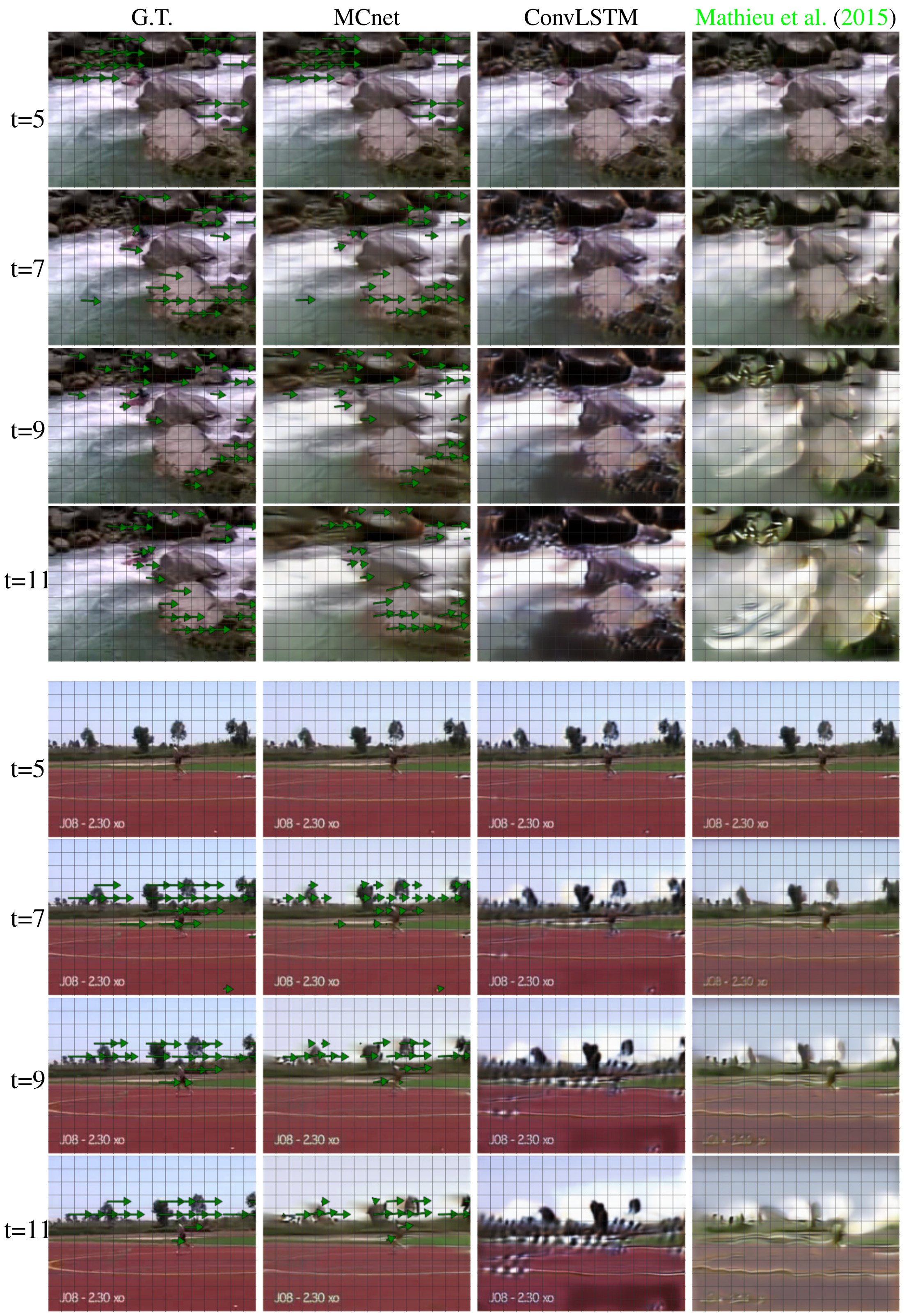


Figure A.5: Qualitative comparisons on UCF-101. We display predictions (in every other frame) starting from the 5th frame. The green arrows denote the top-30 closest optical flow vectors within image patches between MCnet and ground-truth. More clear motion prediction can be seen in the project website.

KTH Results. We were unable to find videos with background motion in the KTH dataset, but we found videos where the camera is zooming in or out for the actions of boxing, handclapping, and handwaving. In Figure A.6, we display qualitative for such videos. Our model is able to predict the zoom change in the cameras, while continuing the action motion. In comparison to the performance observed in UCF101, the background does not change much. Thus, the motion signals are well localized in the foreground motion (human), and do not get confused with the background and lost as quickly.

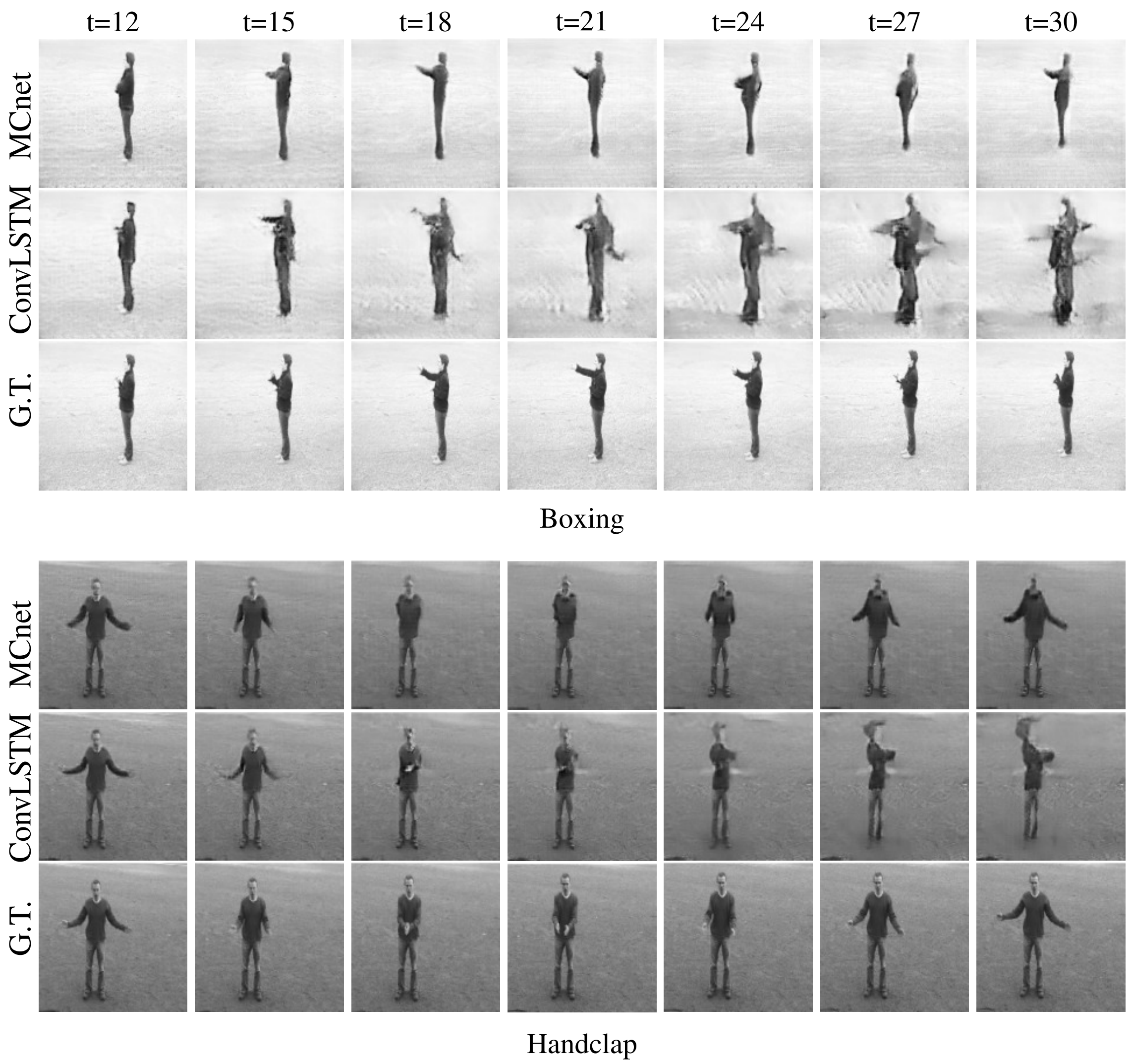


Figure A.6: Qualitative comparisons on KTH testset. We display predictions starting from the 12th frame, in every 3 timesteps. More clear motion prediction can be seen in the project website.

A.2 Extended quantitative evaluation

In this section, we show additional quantitative comparison with a baseline based on copying the last observed frame through time for KTH and UCF101 datasets. Copying the last observed frame through time ensures perfect background prediction in videos where most of the motion comes from foreground (i.e. person performing an action). However, if such foreground composes a small part of the video, it will result in high prediction quality score regardless of the simple copying action.

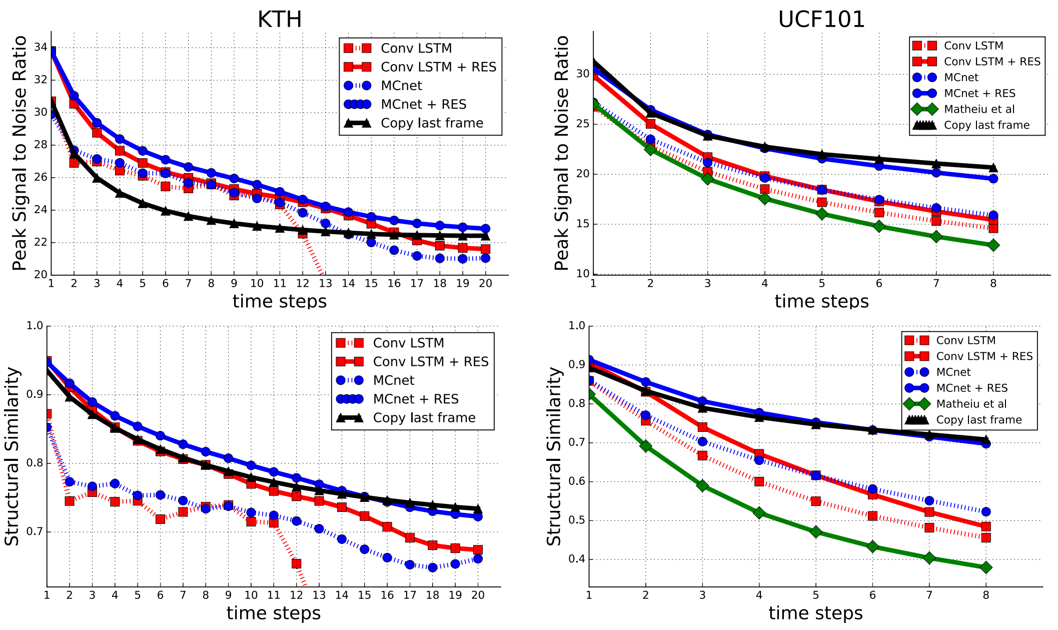


Figure A.7: Extended quantitative comparison including a baseline based on copying the last observed frame through time.

In Figure A.7 below, we can see the quantitative comparison in the datasets. Copying the last observed frame through time does a reasonable job in both datasets, however, the impact is larger in UCF101. Videos in the KTH dataset comprise simple background with minimal camera motion, which allows our network to easily predict both foreground and background motion, resulting in better image quality scores. However, videos in UCF101 contain more complicated and diverse background which in combination with camera motion present a much greater challenge to video prediction networks. From the qualitative results in

Section A.1 and Figures 2.5, A.3, A.4, and A.5, we can see that our network performs better in videos that contain isolated areas of motion compared to videos with dense motion. A simple copy/paste operation of the last observed frame, ensures very high prediction scores in videos where very small motion occur. The considerable score boost by videos with small motion causes the simple copy/paste baseline to outperform MCnet in the overall performance on UCF101.

A.3 UCF101 Motion Disambiguation Experiments

Due to the observed bias from videos with small motion, we perform experiments by measuring the image quality scores on areas of motion. These experiments are similar to the ones performed in Mathieu et al. (2015). We compute DeepFlow optical flow (Weinzaepfel et al., 2013) between the previous and the current groundtruth image of interest, compute the magnitude, and normalize it to $[0, 1]$. The computed optical flow magnitude is used to mask the pixels where motion was observed. We set the pixels where the optical flow magnitude is less than 0.2, and leave all other pixels untouched in both the groundtruth and predicted images. Additionally, we separate the test videos by the average ℓ_2-norm of time difference between target frames. We separate the test videos into deciles based of the computed average ℓ_2-norms, and compute image quality on each decile. Intuitively, the 1^{st} decile contains videos with the least overall of motion (i.e. frames that show the smallest change over time), and the 10^{th} decile contains videos with the most overall motion (i.e. frames that show the largest change over time).

As shown in Figure A.8, when we only evaluate on pixels where rough motion is observed, MCnet reflects higher PSNR and SSIM, and clearly outperforms all the baselines in terms of SSIM. The SSIM results show that our network is able to predict a structure (i.e. textures, edges, etc) similar to the grountruth images within the areas of motion. The PSNR results, however, show that our method outperforms the simple copy/paste baseline for the first few steps, but then our method performs slightly worse. The discrepancies observed between PSNR and SSIM scores could be due to the fact that some of the predicted images may not reflect the exact pixel values of the groundtruth regardless of the structures being similar. SSIM scores are known to take into consideration features in the image that go beyond directly matching pixel values, reflecting more accurately how humans perceived image quality.

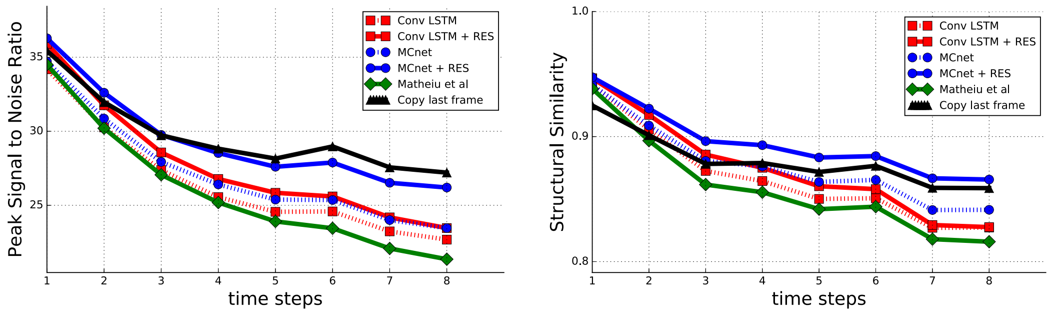


Figure A.8: Extended quantitative comparison on UCF101 including a baseline based on copying the last observed frame through time using motion based pixel mask.

Figures A.10 and A.9 show the evaluation by separating the test videos into deciles based on the average ℓ_2-norm of time difference between target frames. From this evaluation, it is proven that the copy last frame baseline scores higher in videos where motion is the smallest. The first few deciles (videos with small motion) show that our network is not just copying the last observed frame through time, otherwise it would perform similarly to the copy last frame baseline. The last deciles (videos with large motion) show our network outperforming all the baselines, including the copy last frame baseline, effectively confirming that our network does predict motion similar to the motion observed in the video.

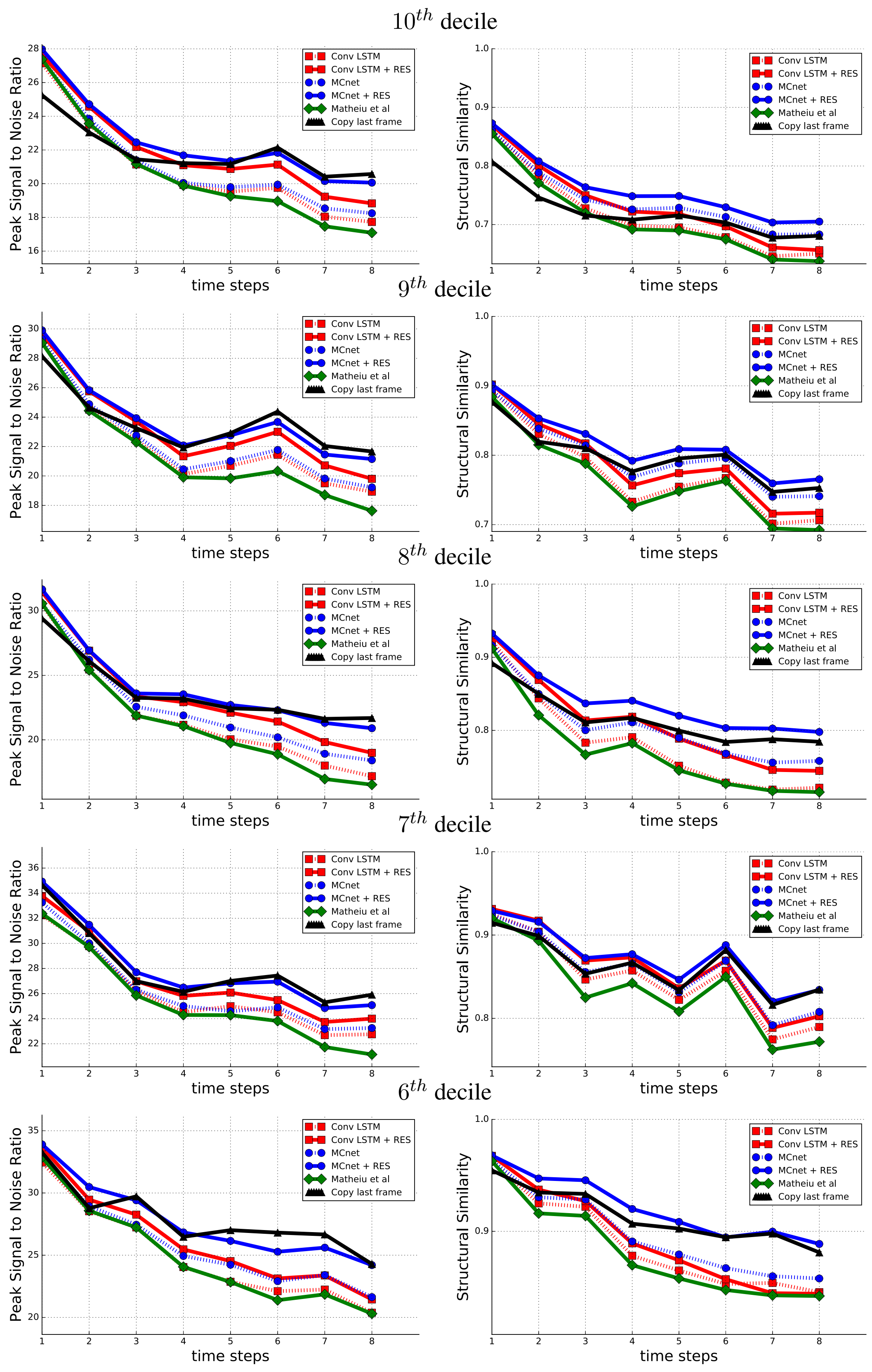


Figure A.9: Quantitative comparison on UCF101 using motion based pixel mask, and separating dataset by average ℓ_2-norm of time difference between target frames.

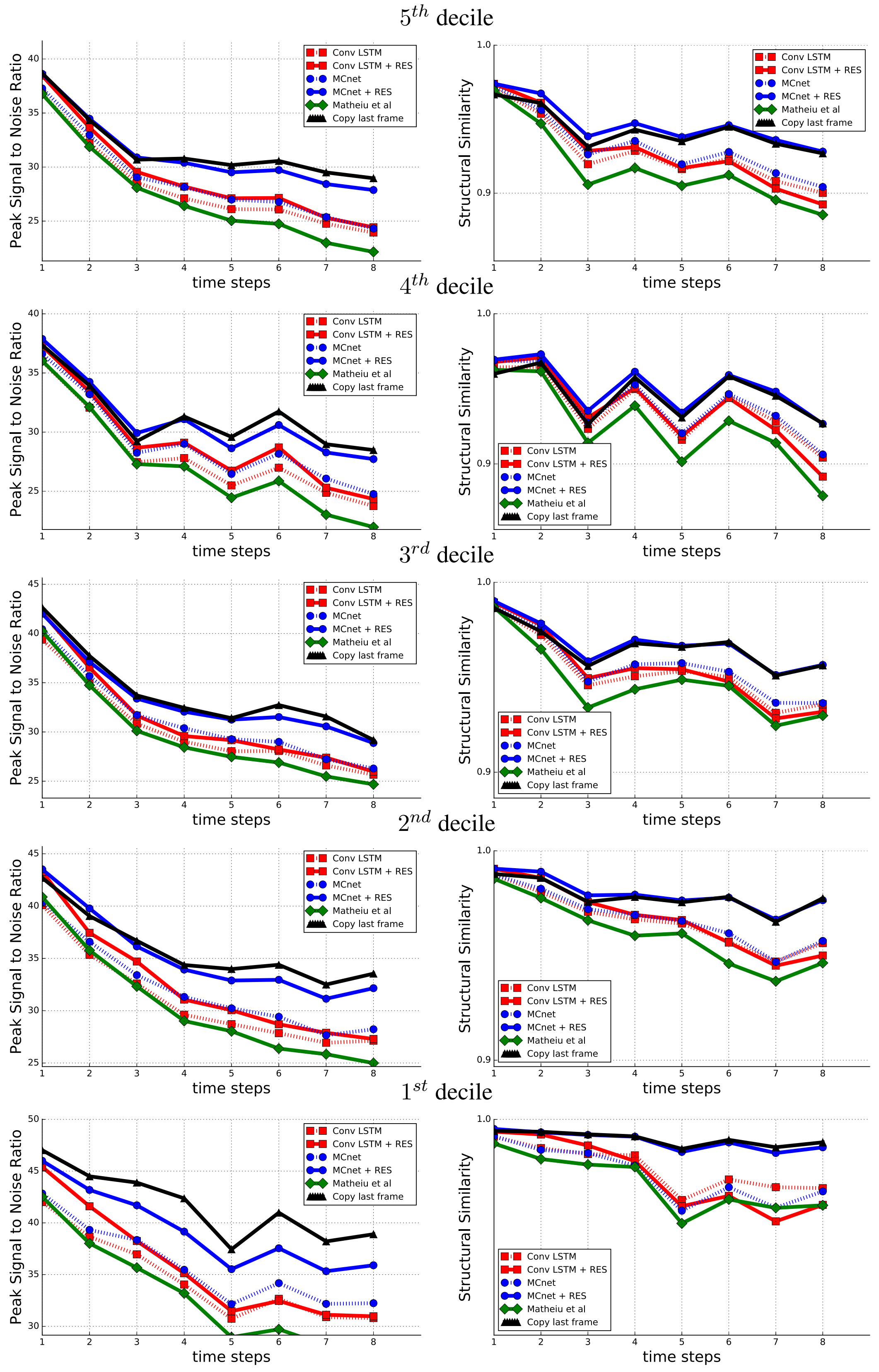


Figure A.10: Quantitative comparison on UCF101 using motion based pixel mask, and separating dataset by average ℓ_2-norm of time difference between target frames.

APPENDIX B

Learning to Generate Long-term Future via Hierarchical Prediction

B.1 Motion-Based Pixel-Level Evaluation, Analysis, and Control Experiments

In this section, we evaluate the predictions by deciles of motion similar to Villegas et al. (2017a) using Peak Signal-to-Noise Ratio (PSNR) measure, where the 10th decile contains videos with the most overall motion. We add a modification to our hierarchical method based on a simple heuristic by which we copy the background pixels from the last observed frame using the predicted pose heat-maps as foreground/background masks (`Ours BG`). Additionally, we perform experiments based on an *oracle* that provides our image generator the exact future pose trajectories (`Ours GT-pose`*) and we also apply the previously mentioned heuristics (`Ours GT-pose BG`*). We put * marks to clarify that these are *hypothetical* methods as they require ground-truth future pose trajectories.

In our method, the future frames are strictly dictated by the future structure. Therefore, the prediction based on the future pose oracle sheds light on how much predicting a different future structure affects PSNR scores. (Note: many future trajectories are possible given a single past trajectory.) Further, we show that our conditional image generator given the perfect knowledge of the future pose trajectory (e.g., `Ours GT-pose`*) produces high-quality video prediction that both matches the ground-truth video closely and achieves much higher PNSRs. These results suggest that our hierarchical approach is a step in the right direction towards solving the problem of long-term pixel-level video prediction.

B.1.1 Penn Action

In Figures B.1, and B.2, we show evaluation on each decile of motion. The plots show that our method outperforms the baselines for long-term frame prediction. In addition, by using the future pose determined by the oracle as input to our conditional image generator, our method can achieve even higher PSNR scores. We hypothesize that predicting future frames that reflect similar action semantics as the ground-truth, but with possibly different pose trajectories, causes lower PSNR scores. Figure B.3 supports this hypothesis by showing that higher MSE in predicted pose tends to correspond to lower PSNR score.

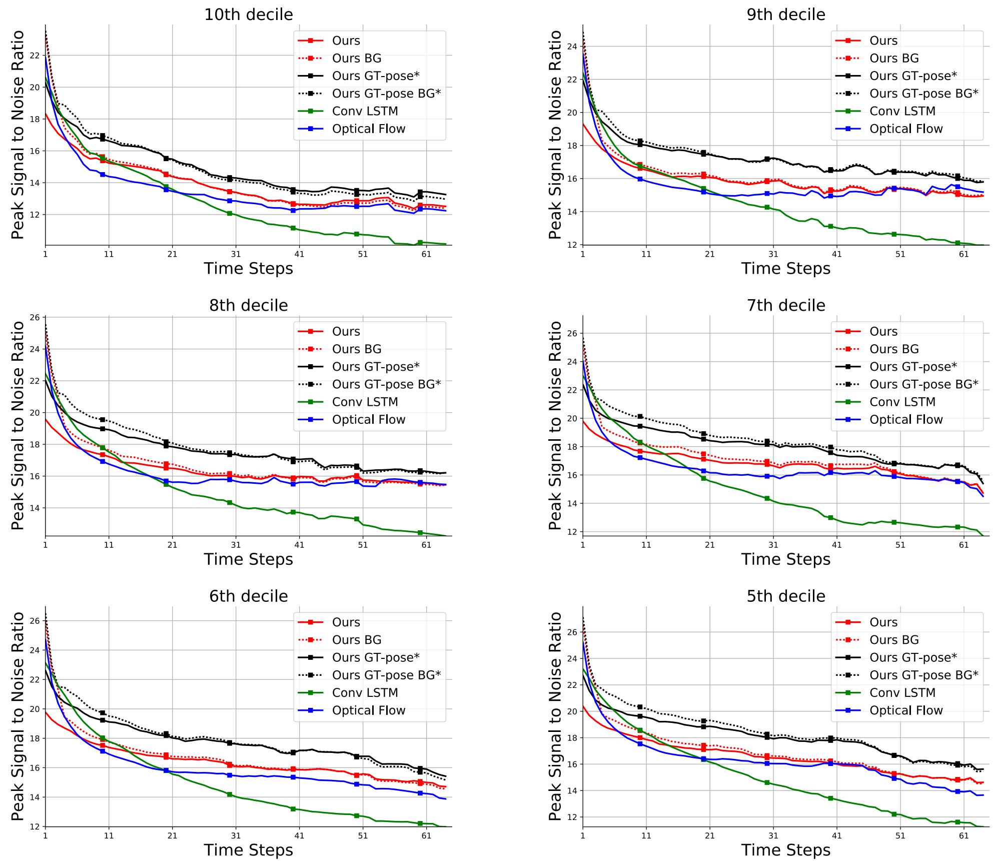


Figure B.1: Quantitative comparison on Penn Action separated by motion decile.

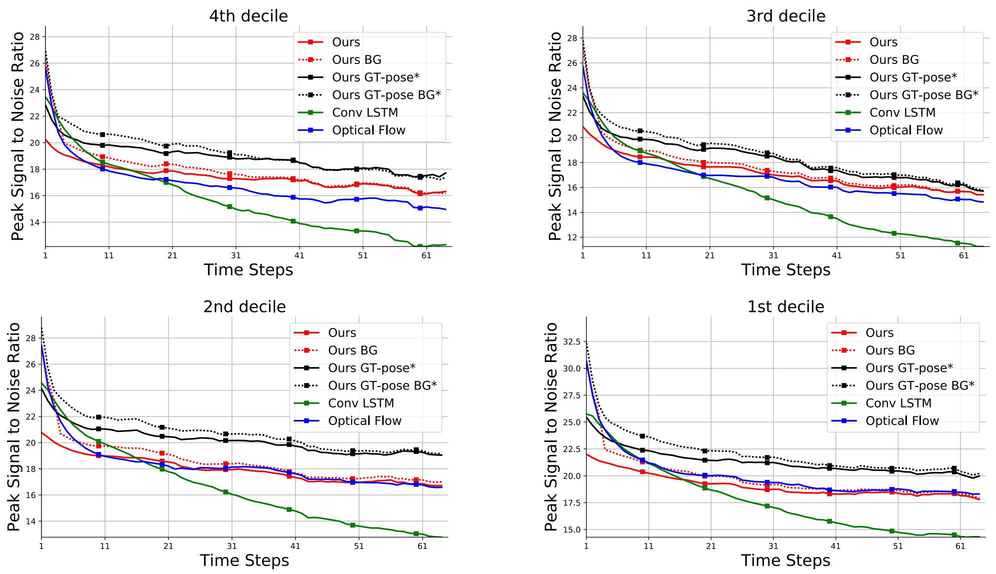


Figure B.2: (Continued from Figure B.1.) Quantitative comparison on Penn Action separated by motion decile.

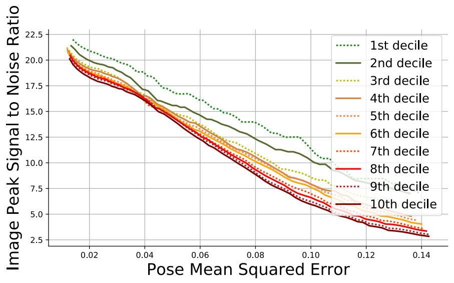
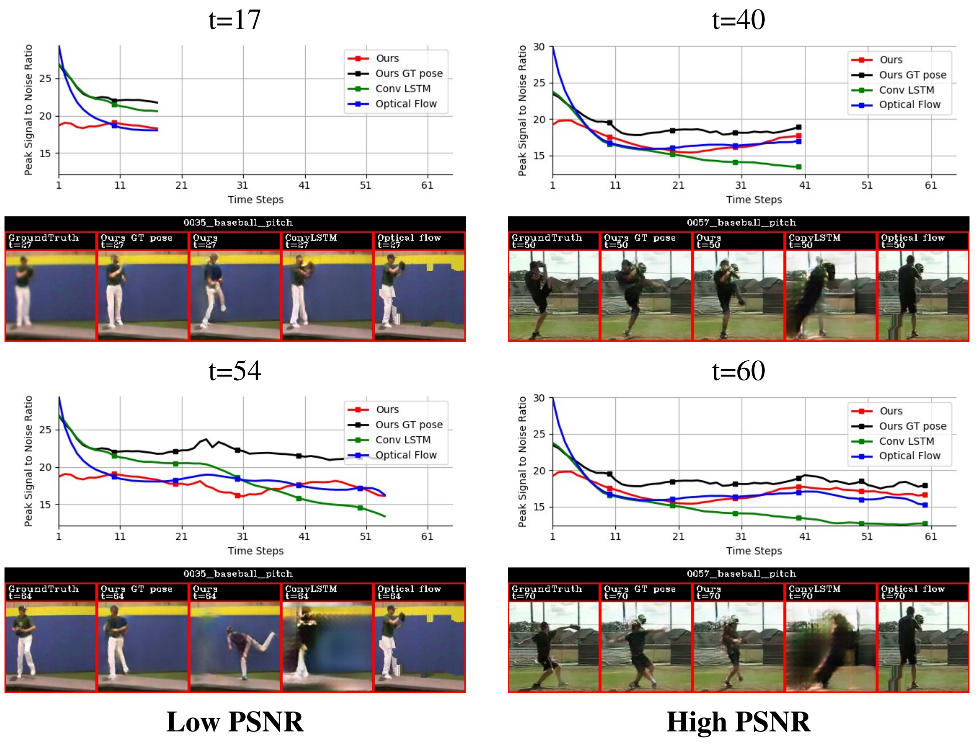


Figure B.3: Predicted frames PSNR vs. Mean Squared Error on the predicted pose for each motion decile in Penn Action.

The fact that PSNR can be low even if the predicted future is one of the many plausible futures suggest that PSNR may not be the best way to evaluate long-term video prediction when only a single future trajectory is predicted. This issue might be alleviated when a model can predict multiple possible future trajectories, but this investigation using our hierarchical decomposition is left as future work. In Figures B.4 and B.5, we show videos where PSNR is low when a different future (from the ground-truth) is predicted (left), and video where PSNR is high because the predicted future is close to the ground-true future (right).



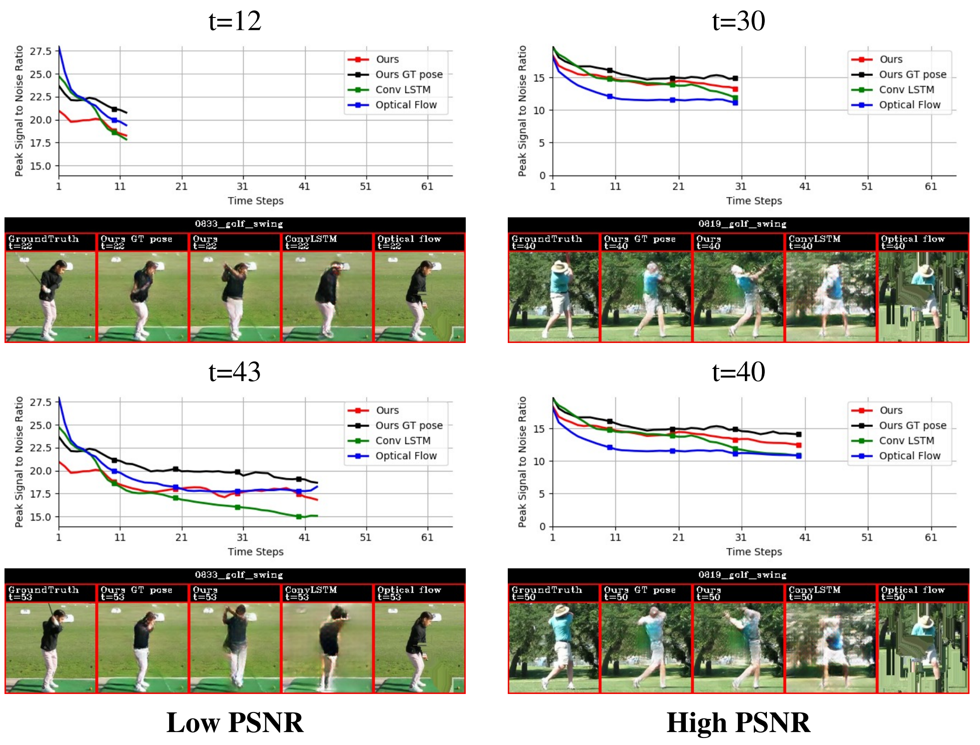


Figure B.4: Quantitative and visual comparison on Penn Action for selected time-steps for the action of `baseball pitch` (top) and `golf swing` (bottom). Side by side video comparison can be found in our project website

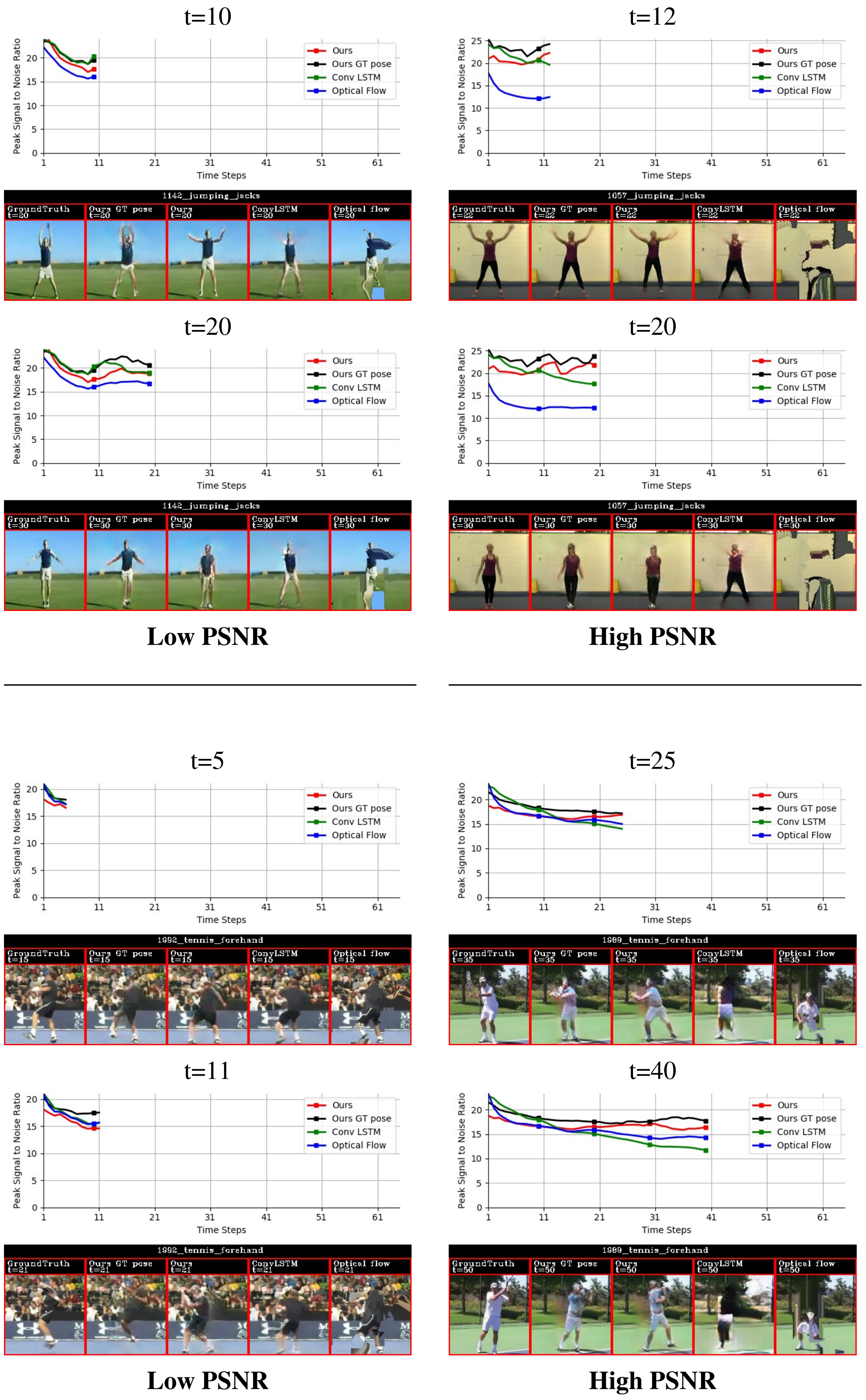


Figure B.5: Quantitative and visual comparison on Penn Action for selected time-steps for the actions of `jumping jacks` (top) and `tennis forehand` (bottom). Side by side video comparison can be found in our project website

To directly compare our image generator using the predicted future pose (`Ours`) and the ground-truth future pose given by the oracle (`Ours GT-pose`*), we present qualitative experiments in Figure B.6 and Figure B.7. We can see that the both predicted videos contain the action in the video. The oracle based video prediction reflects the exact future very well.

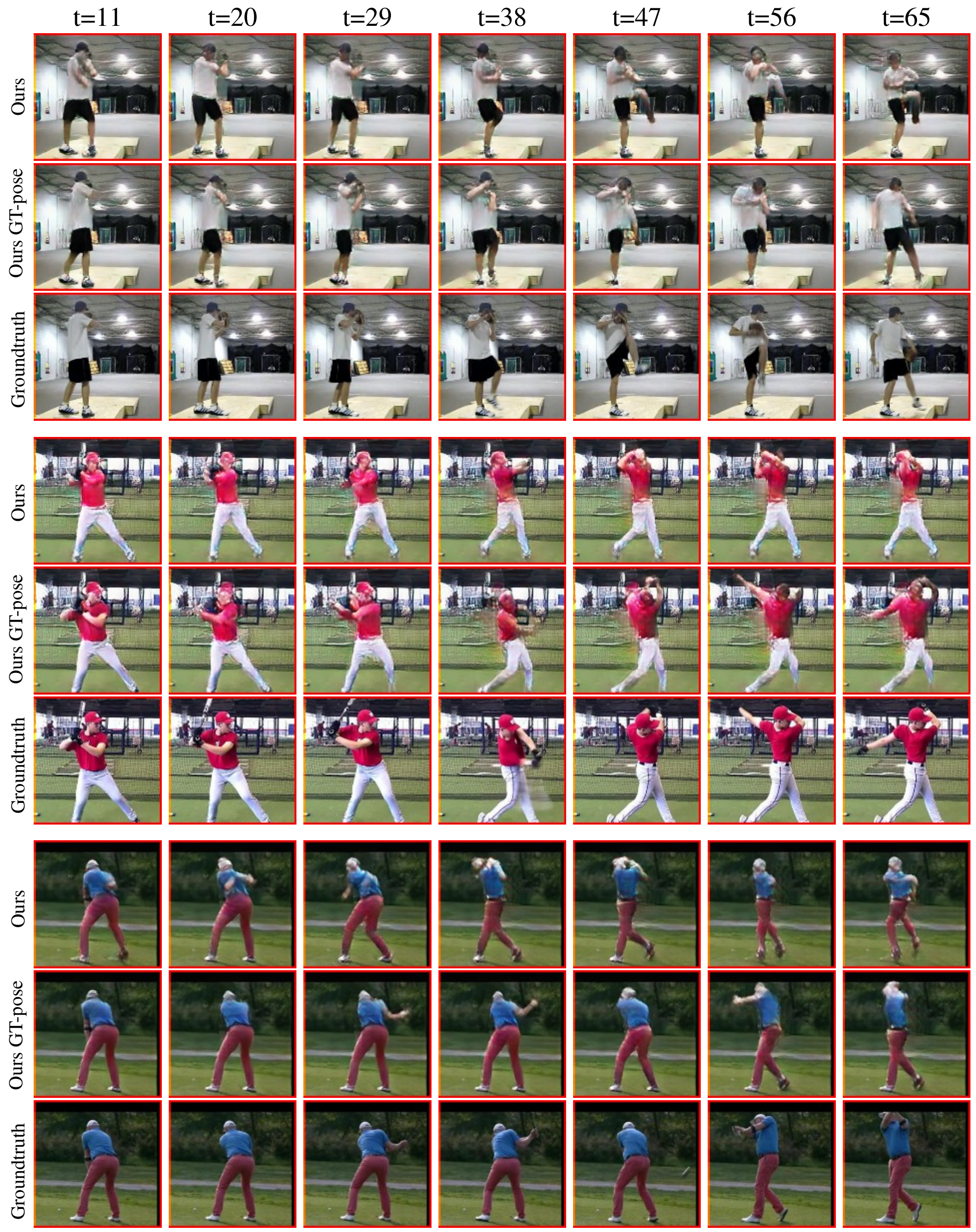


Figure B.6: Qualitative evaluation of our network for long-term pixel-level generation. We show the actions of `baseball pitch` (top row), `baseball swing` (middle row), and `gold swing` (bottom row). Side by side video comparison can be found in our project website.

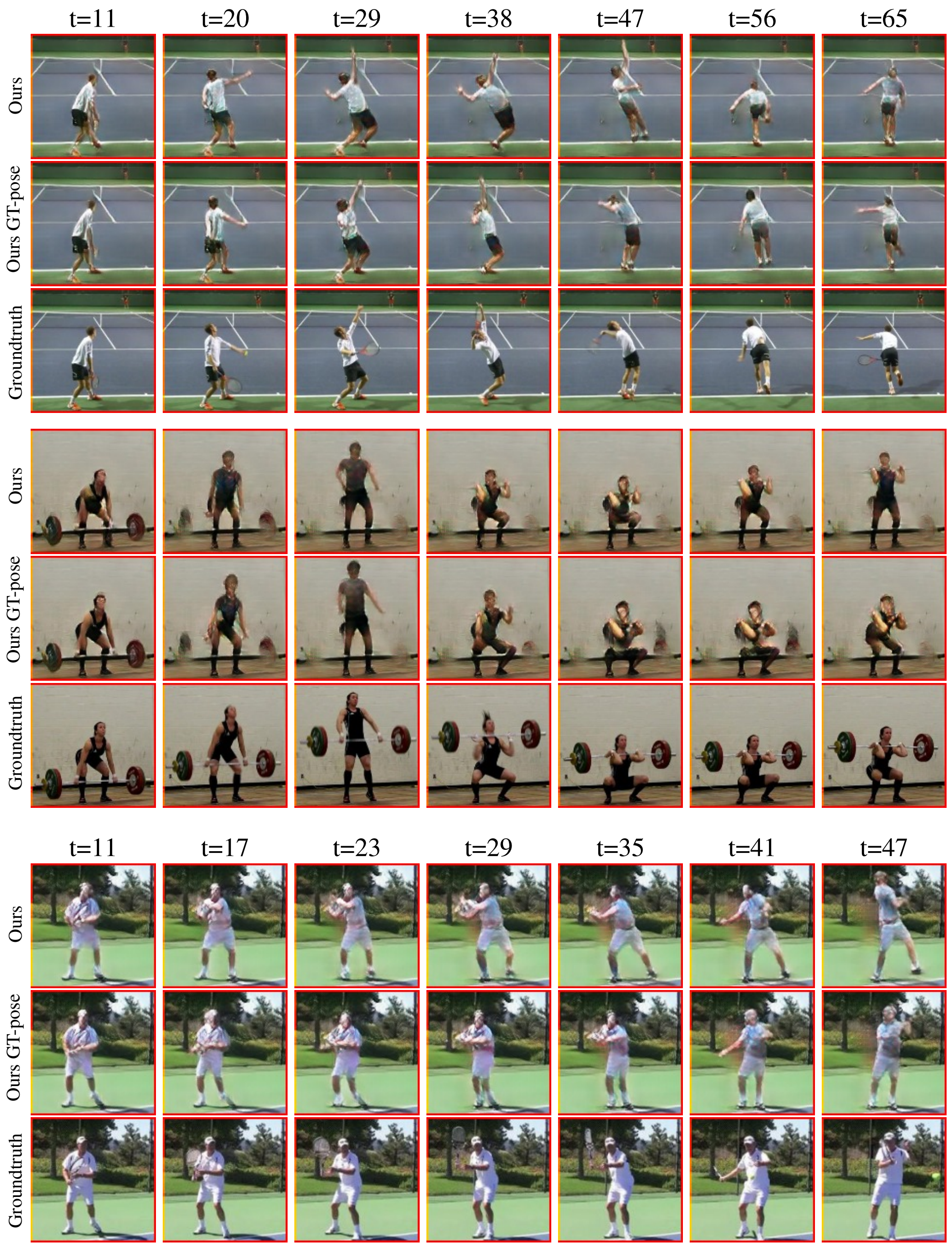


Figure B.7: Qualitative evaluation of our network for long-term pixel-level generation. We show the actions of `tennis serve` (top row), `clean and jerk` (middle row), and `tennis forehand` (bottom row). We show a different timescale for `tennis forehand` because the ground-truth action sequence does not reach time step 65. Side by side video comparison can be found in our project website.

B.1.2 Human3.6M

In Figure B.8, we show evaluation (PSNRs over time) of different methods on each decile of motion.

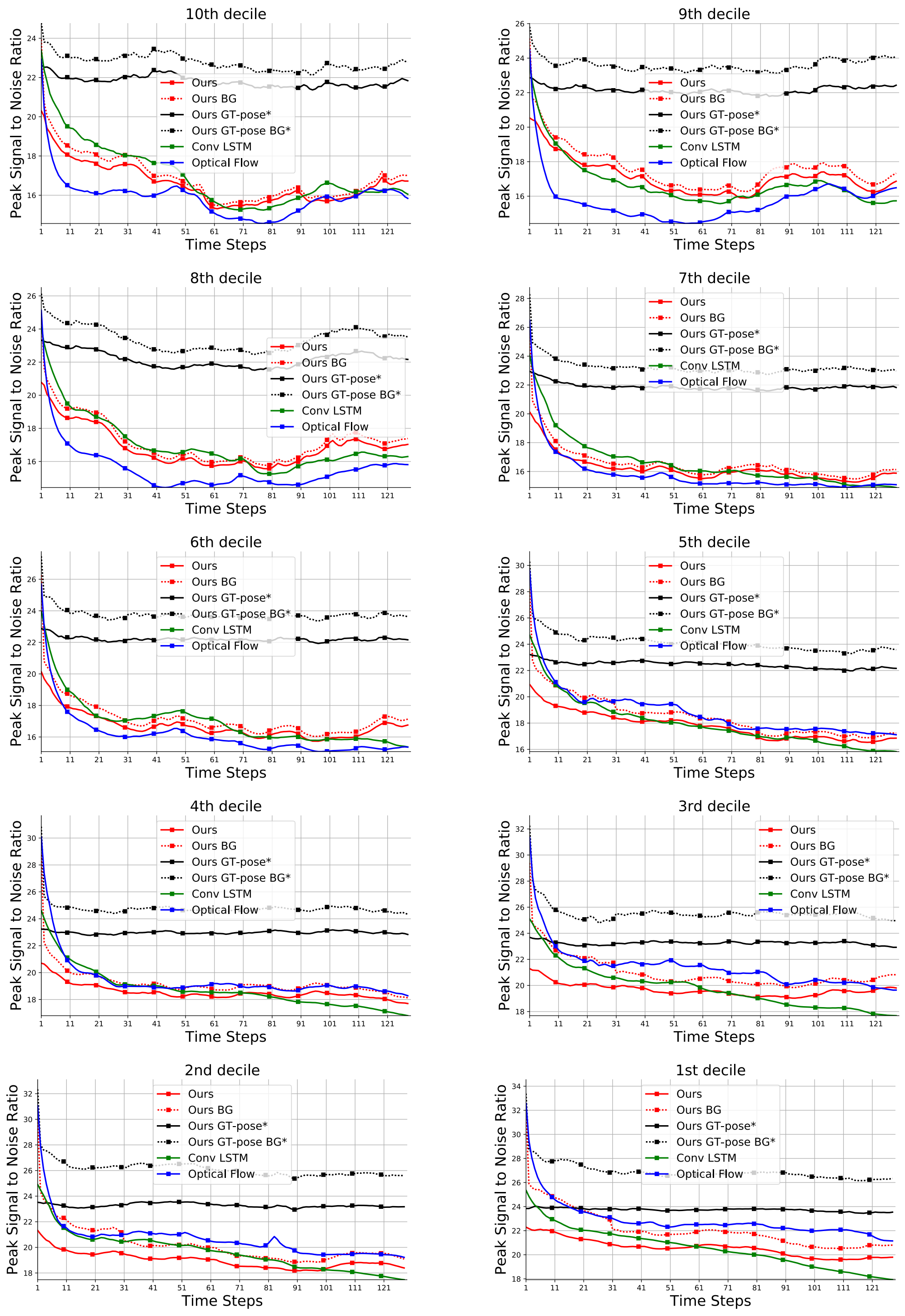


Figure B.8: Quantitative comparison on Human3.6M separated by motion decile.

As shown in Figure B.8, our hierarchical approach (e.g., `Ours BG`) tends to achieve PSNR performance that is better than optical flow based method and comparable to convolutional LSTM. In addition, when using the oracle future pose predictor as input to our image generator, the PSNR scores get a larger boost compared to Section B.1.1. This is because there is higher uncertainty of the actions being performed in the Human 3.6M dataset compared to Penn Action dataset. Therefore, even plausible future predictions can still deviate significantly from the ground-truth future trajectory, which can penalize PSNRs.

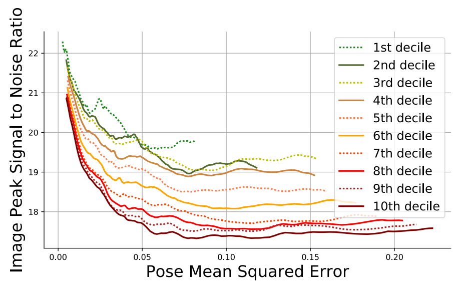


Figure B.9: Predicted frames PSNR vs. Mean Squared Error on the predicted pose for each motion decile in Human3.6M.

To gain further insight on this problem, we provide two additional analysis. First, we compute how the average PSNR changes as the future pose MSE increases in Figure B.9. The figure clearly shows the negative correlation between the predicted pose MSE and frame PSNR, meaning that larger deviation of the predicted future pose from the ground future pose tend to cause lower PSNRs.

Second, we show snapshots of video prediction from different methods along with the PNSRs that change over time (Figures B.11 and B.10). Our method tend to make plausible future pose trajectory but it can deviate from the ground-truth future pose trajectory; in such case, our method tend to achieve low PSNRs. However, when the future pose prediction from our method matches well with the ground-truth, the PSNR is much higher and the generated image frame is perceptually very similar to the ground-truth frame. In contrast, optical flow and convolutional LSTM make prediction that often loses the structure of the foreground (e.g., human) over time, and eventually their predicted videos tend to become *static*. It is interesting to note that our method is comparable to convolutional LSTM in terms of PSNR, but that our method still strongly outperforms convolutional LSTM in terms of human evaluation, as described in Section 3.6.2.

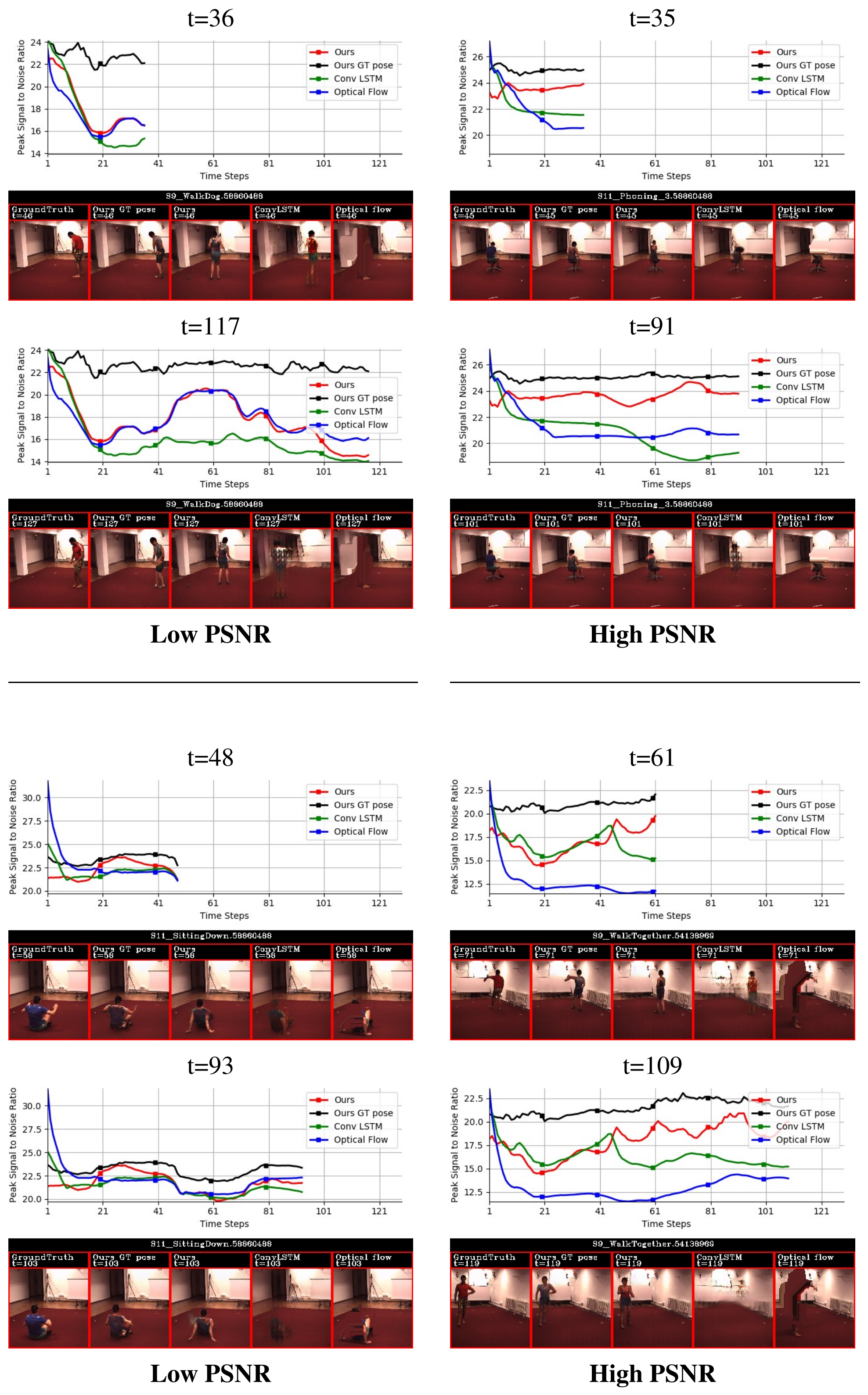


Figure B.10: Quantitative and visual comparison on Human 3.6M for selected time-steps for the actions of `walk dog` (top left), `phoning` (top right), `sitting down` (bottom left), and `walk together` (bottom right). Side by side video comparison can be found in our project website.

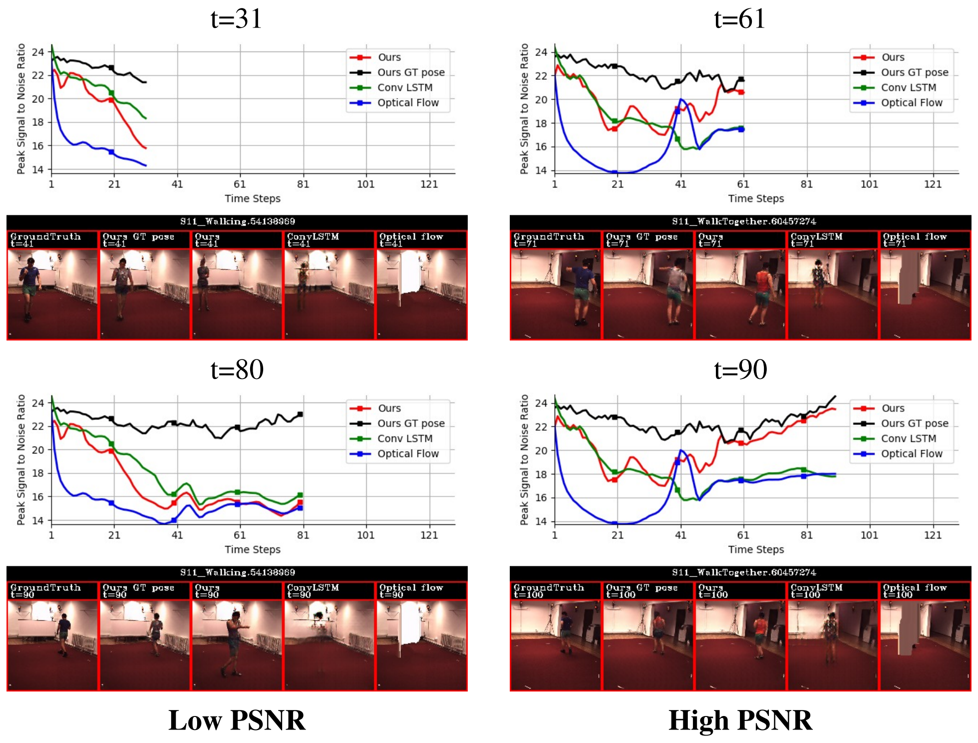


Figure B.11: Quantitative and visual comparison on Human 3.6M for selected time-steps for the action of `walking` (left) and `walk together` (right). Side by side video comparison can be found in our project website.

To directly compare our image generator using the predicted future pose (`Ours`) and the ground-truth future pose given by the oracle (`Ours GT-pose*`), we present qualitative experiments in Figure B.12 and Figure B.13. We can see that the both predicted videos contain the action in the video. However, the oracle based video reflects the exact future very well.

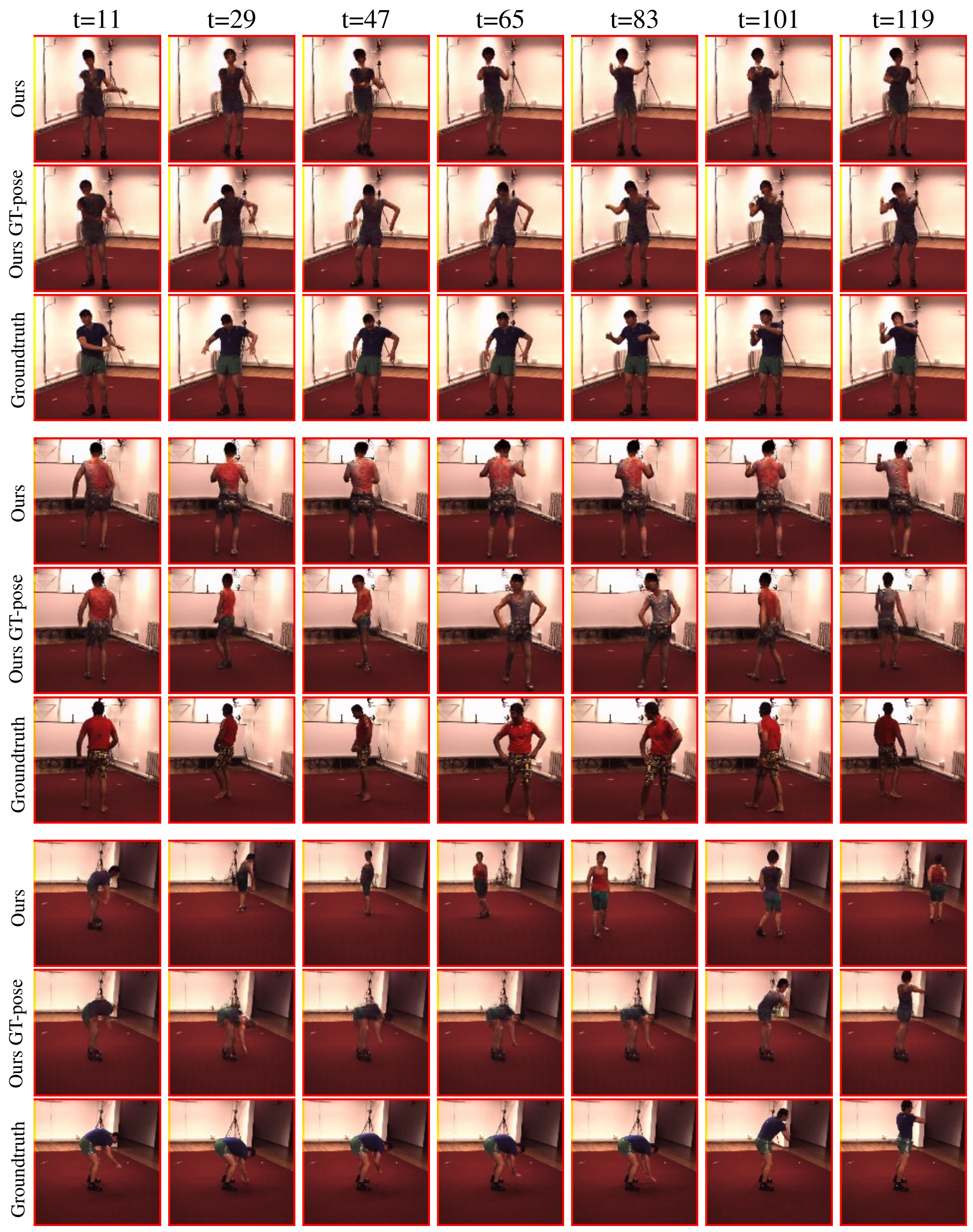


Figure B.12: Qualitative evaluation of our network for long-term pixel-level generation. We show the actions of `giving directions` (top three rows), `posing` (middle three rows), and `walk dog` (bottom three rows). Side by side video comparison can be found in our project website.

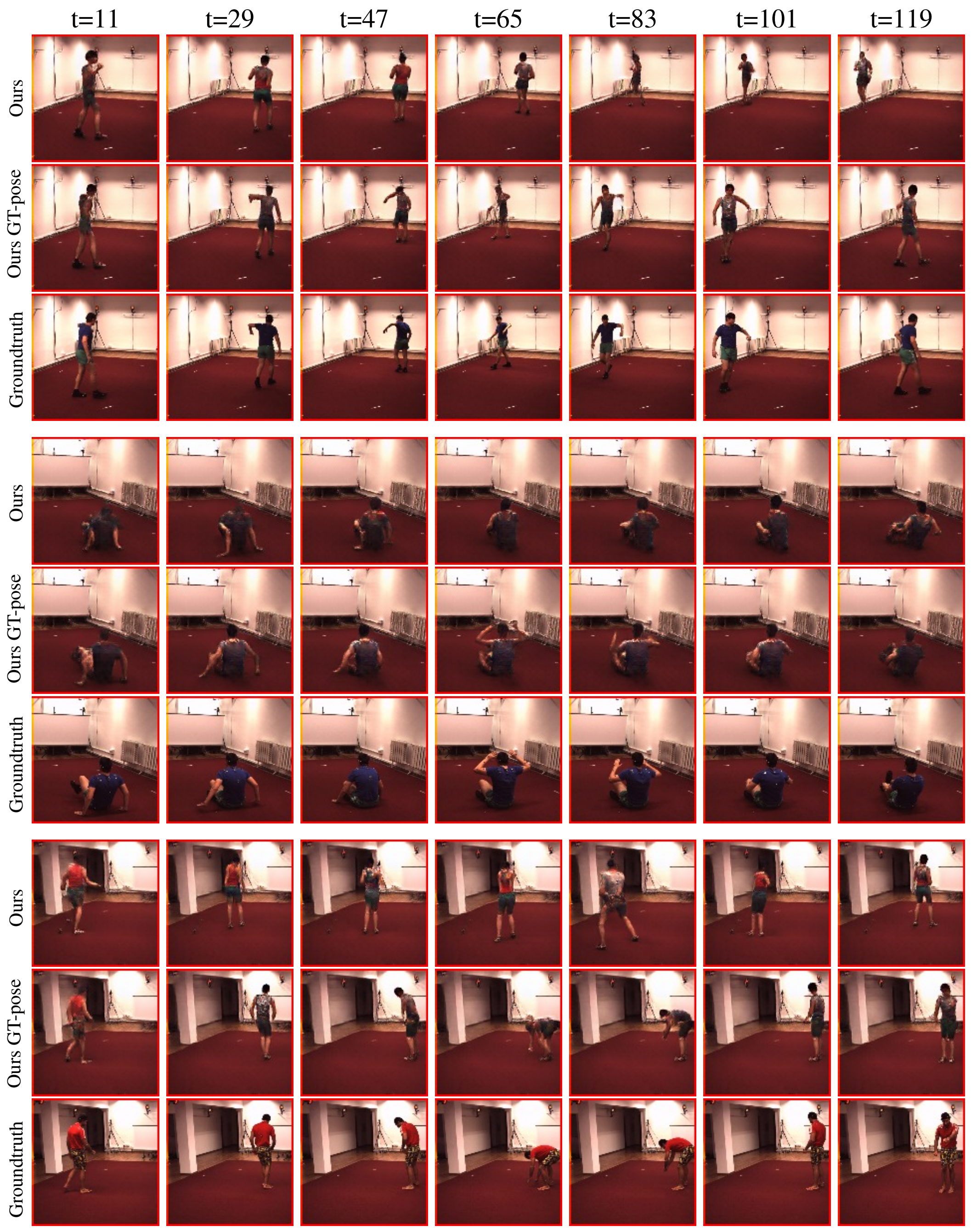


Figure B.13: Qualitative evaluation of our network for long-term pixel-level generation. We show the actions of `walk together` (top three rows), `sitting down` (middle three rows), and `walk dog` (bottom three rows). Side by side video comparison can be found in our project website.

APPENDIX C

High Fidelity Video Prediction with Large Neural Nets

C.0.1 Video results

We have provided video comparisons of the baseline and largest model for the best two models (LSTM and SVG') in this website: https://cutt.ly/QGuCex.

C.0.2 Per-frame evaluation comparison as model capacity increases

In this section, we present a per-frame evaluation for capacities in each of the models we experiment in our paper.

C.0.2.1 Robot Arm.

The plots show a slight improvement as the number of parameters increase for the CNN architecture. However, for the LSTM and SVG' architectures the improvement is more noticeable. We hypothesize that this is due to the model being able to better handle the robot arm interaction with the objects by having a large capacity.

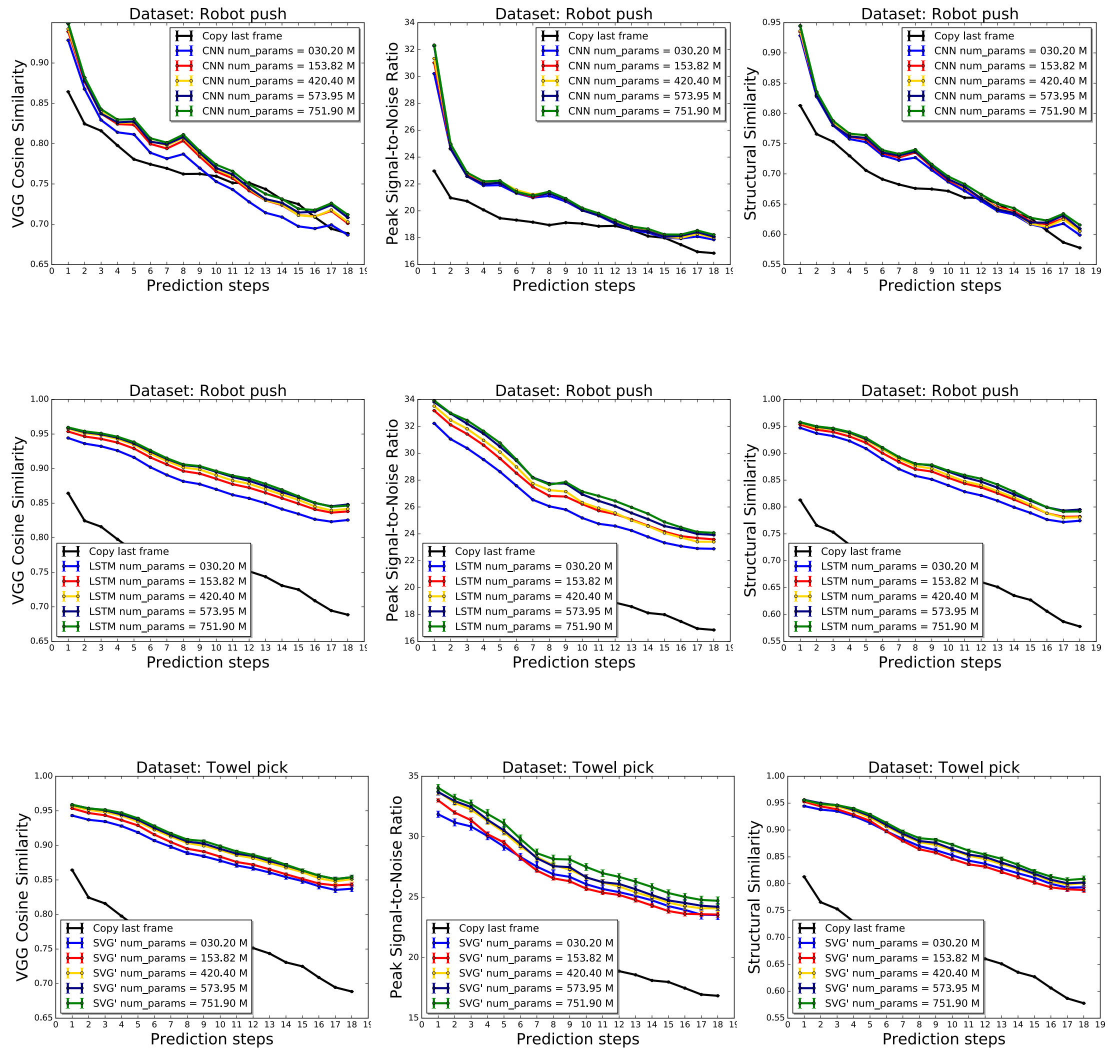


Figure C.1: Towel pick per-frame evaluation (higher is better). As capacity increases, the per frame evaluation metrics become better. The increase is due to better modeling of interactions. The objects become sharper, and robot arm dynamics become better as the model capacity increases.

C.0.2.2 Human Activities.

The Human 3.6M dataset is mostly made of static background and the moving human occupies a relatively very small area of the frame. Therefore, models that are not capable of perfectly predicting the background become affected by this. To show our point, we include a baseline where we simply copy the last observed frame through time. This baseline significantly outperforms all models. Therefore, from these results we can conclude that per-frame evaluations are not reliable when a large portion of a video does not move.

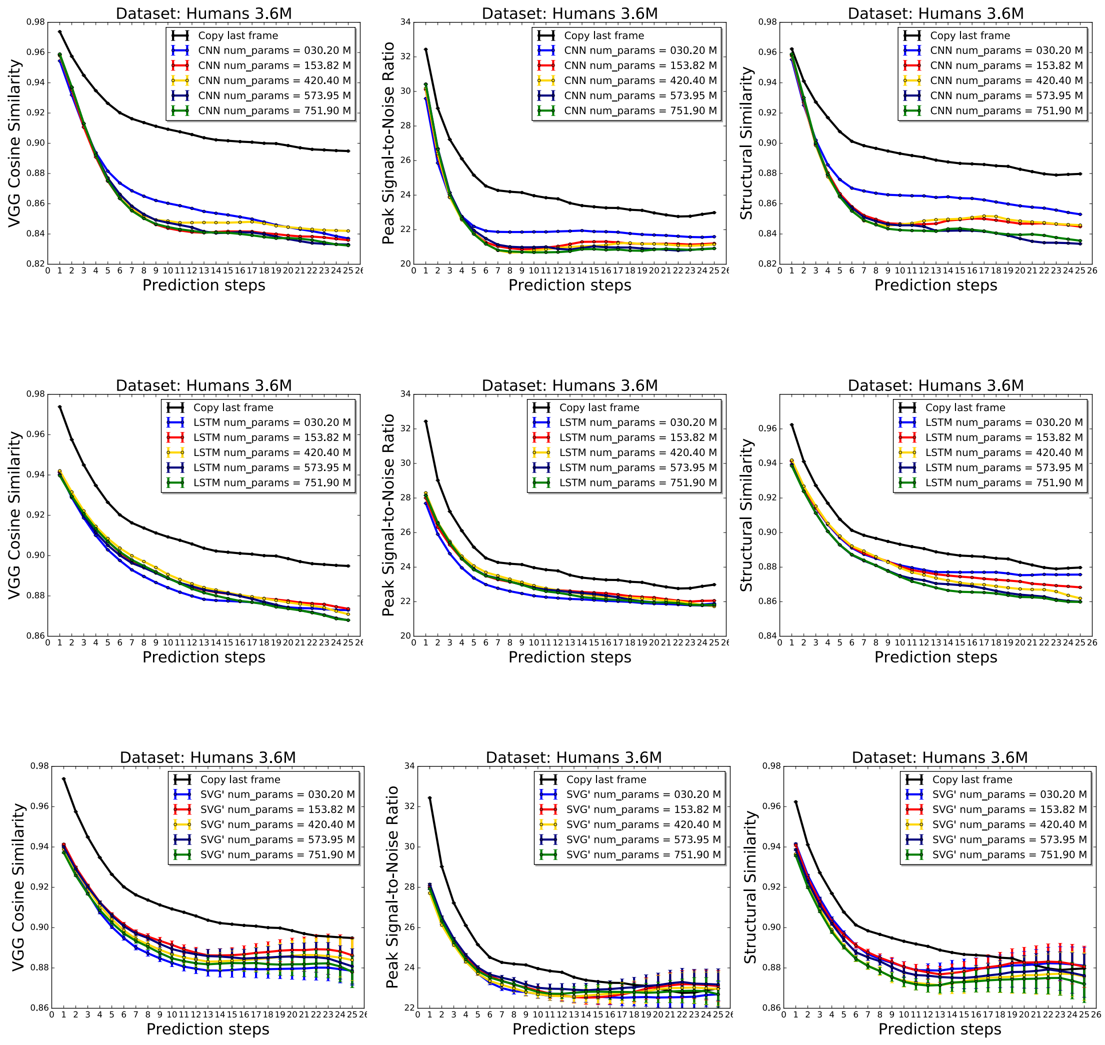


Figure C.2: Human 3.6M per-frame evaluation (higher is better). In this dataset, there is a large amount of non-moving background that causes a per-frame evaluation to become not reliable. This is shown by the baseline based on simply copying the last observed frame through time which significantly outperforms all methods.

C.0.2.3 Car Driving.

In this dataset, as observed by the FVD measure in the main text, we see that the CNN model fails to make improvement in the per-frame evaluation metrics. However, the LSTM and SVG' models performance improves as the capacity of the models increases.

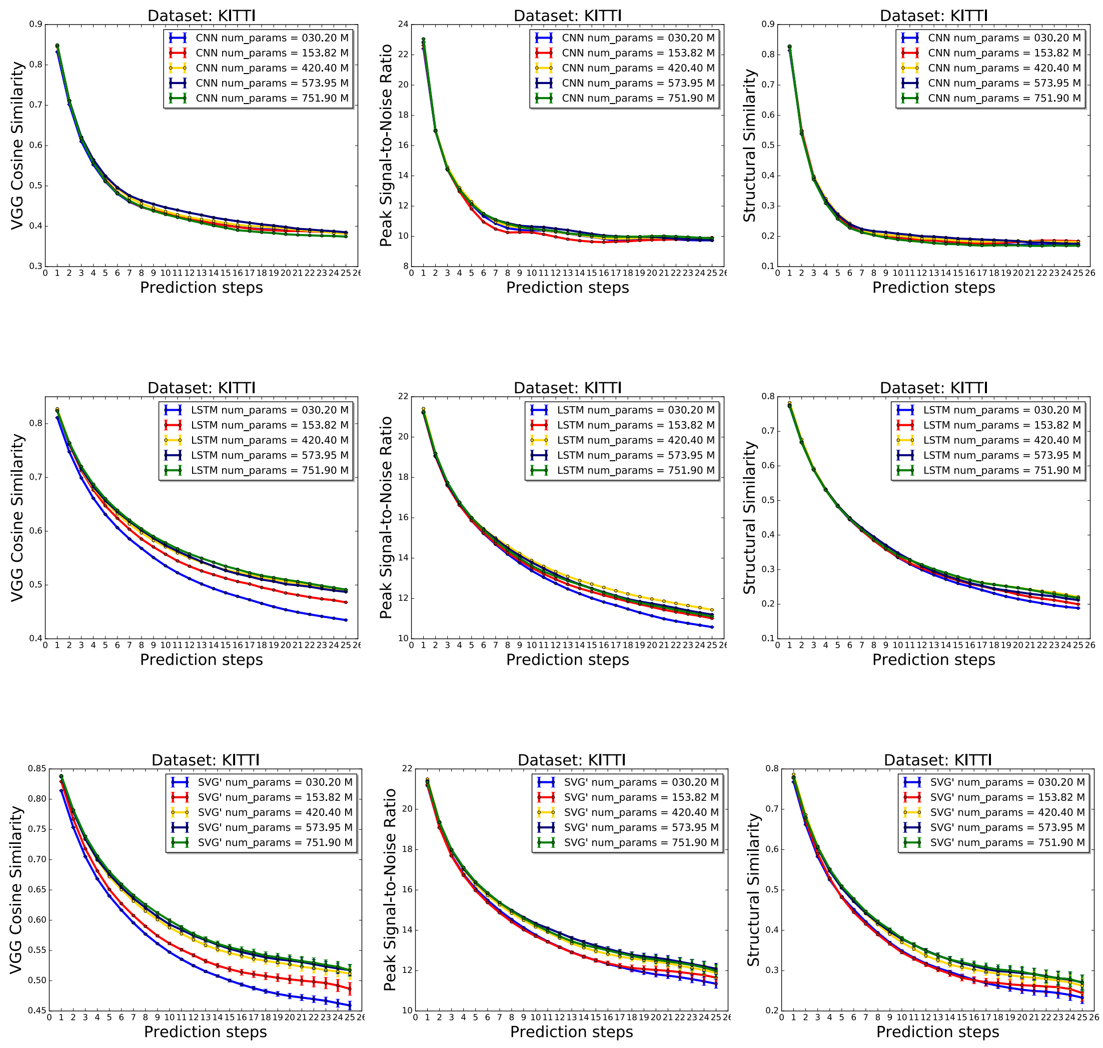


Figure C.3: KITTI driving per-frame evaluation (higher is better). As capacity increases, the per frame evaluation metrics become better. The increase is due to better modeling the driving dynamics and partial observability. Due to the difficulty of predicting the exact not-observed parts of the image, the performance converges toward the largest models.

The metric in which this is the most obvious is the VGG Cosine Similarity. This may be due to the partial observability of the dataset which makes it very difficult to predict exact pixels into the future, and so, PSNR and SSIM do not result in a large gap between the larger and baseline models. However, VGG Cosine Similarity compares high-level features of the predicted frames. Therefore, even if the predicted pixels are not exact, the predicted structures in the frames may be similar to those the ground-truth future. For this dataset, we do not present a copy last frame baseline because most pixels move (in contrast to the robot arm and Human 3.6M dataset, where many pixels stay fixed).

C.0.3 Effects of using skip connections in video prediction

In this section, we present a study on the effects of using skip connections from encoder to decoder. Similar to Denton and Fergus (2018), the method presented in the main text has skip connections going from the encoder of the last observed frame directly to the decoder for all frame predictions. This allows the video prediction method to choose to transfer pixels that did not move from the input frame directly into the output frame, and generate the pixels that move. Below, we show the performance for each of the datasets presented in this work.

C.0.3.1 Robot Arm.

In Figure C.4, we can see that skip connections do play an important role in terms of FVD evaluation for the robot arm action conditioned experiments. This implies that having skip connections eases the difficulty of video prediction in that it is only required to model the dynamics of the moving parts and everything else can simply be transferred to the output frames.

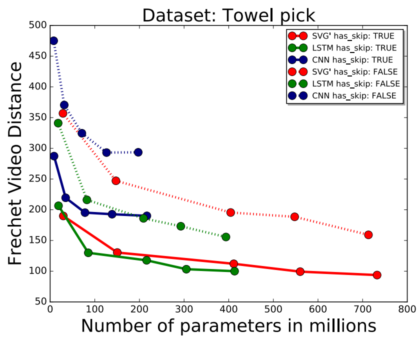


Figure C.4: Towel pick video dynamics evaluation (lower is better). Solid lines define method with skip connections and dotted lines without skip connections.

In addition, having skip connections also help to make more accurate frame-wise predictions. In Figure C.5, the advantage of having skip connections is clear in all prediction steps. This indicates that skip connections are not just essential for predicting dynamics that look like the ground-truth videos, but also, the accuracy of the predicted pixels becomes better.

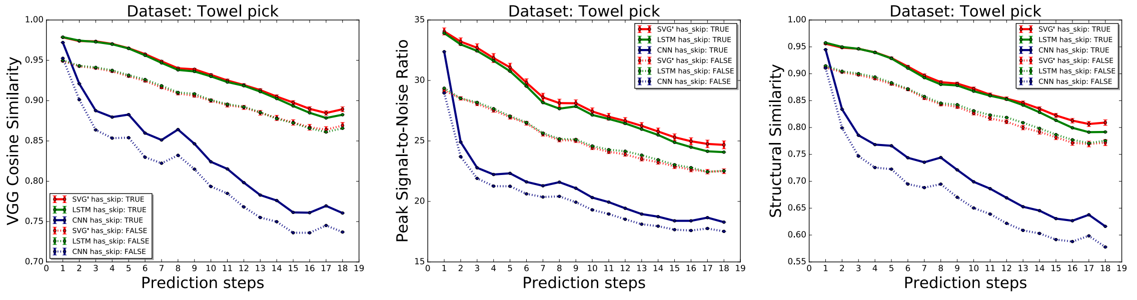


Figure C.5: Towel pick per-frame evaluation (higher is better). Solid lines define method with skip connections and dotted lines without skip connections.

C.0.3.2 Human Activities.

In Figure C.6, having skip connections results in a large performance improvement in FVD for the CNN based video prediction architecture. However, for the LSTM and SVG' based architectures, we can that there is not clear improvement as the model size increases. We hypothesize that, since there are no interactions, the background is static, and the background between training and testing data is similar, the dataset dynamics become easier to model. Therefore, there is no need for the model to separate moving and non-moving parts to achieve good predictions.

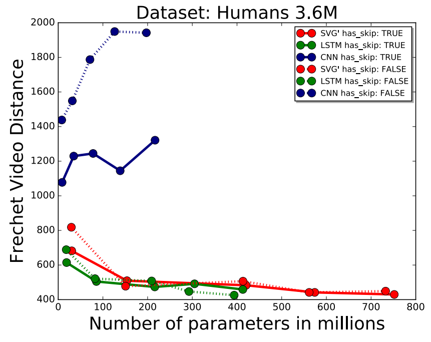


Figure C.6: Human 3.6M video dynamics evaluation (lower is better). Solid lines define method with skip connections and dotted lines without skip connections.

In contrast to FVD evaluation, having skip connections greatly improves the performance in the per-frame evaluation metrics for all models (Figure C.7). This is mainly due to the fact that the moving humans take up a very small portion of the image. Thus, having a way to transfer non-moving pixels directly into the output frames results in more accurate per-frame performance.

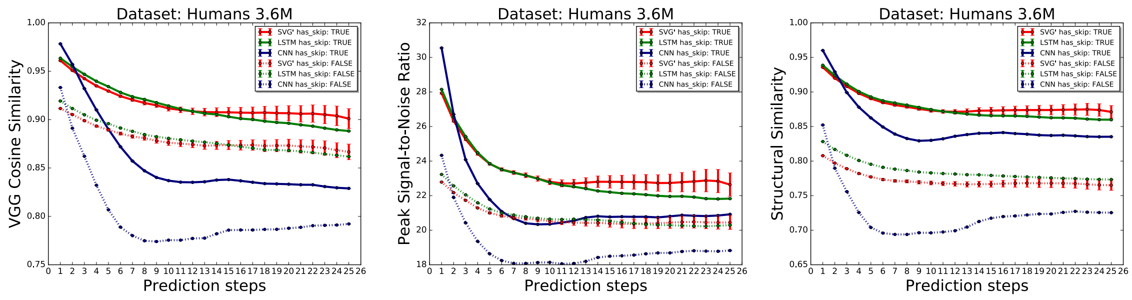


Figure C.7: Human 3.6M per-frame evaluation (higher is better). Solid lines define method with skip connections and dotted lines without skip connections.

C.0.3.3 KITTI driving.

In Figure C.8, we can see that for the recurrent models (LSTM and SVG') having skip connections results in improved FVD performance. However, when using a CNN based architecture, is clear for most models, but not all of them as the two curves become close to each other when M and K are make the model twice and three times bigger than the original model (second and third parameter value in the x-axis). We hypothesize that this happens because almost all pixels move in these videos, and so, simple skip connections without recurrent steps to remember what pixels are moving throughout the prediction makes skip connections not as critical for the intermediate size models.

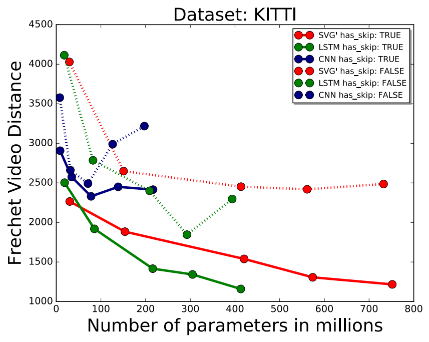


Figure C.8: KITTI driving video dynamics evaluation (lower is better). Solid lines define method with skip connections and dotted lines without skip connections.

In terms of per-frame evaluation, we see an interesting behavior as prediction move forward in time (Figure C.9). The predicted frames become less accurate as time moves forward; effectively reducing the performance gap between the architectures with and without skip connections. This happens because predicting videos in this dataset requires predicting unseen pixels moving into view (e.g., partial observability). Therefore, having skip connections can only help for predicting nearby frames and eventually requires generating fully unseen objects in the frames. The probability that the exact pixels are generated reduces as time moves forward, even if the overall predicted dynamics are within what is realistic in the dataset.

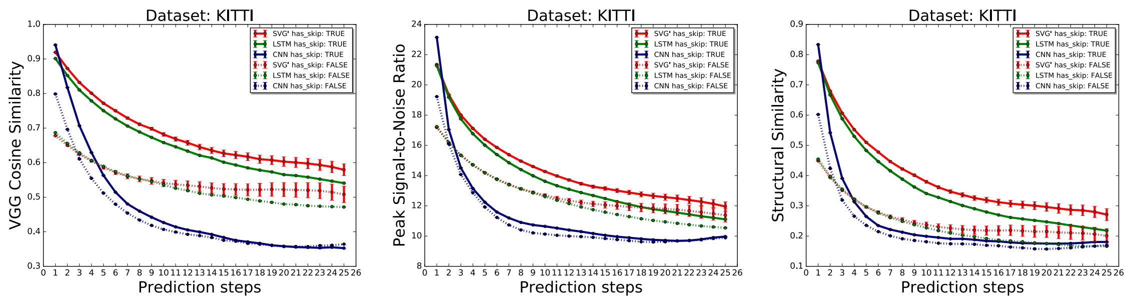


Figure C.9: KITTI driving per-frame evaluation (higher is better). Solid lines define method with skip connections and dotted lines without skip connections.

C.0.4 All-vs-all Amazon Mechanical Turk comparison

In this section, we compare the largest models we trained for the different inductive bias considered in our study. Similar to the experiments presented in the may text, we use 10 unique workers per video and choose the selection with the most votes as the final answer. The videos used in the comparison are determined by the highest VGG Cosine Similarity score amongst all samples for the stochastic model, and we use the single trajectory produced by LSTM and CNN.

Dataset	Method 1	Method 2	Method 1	Method 2	About the same
Towel Pick	SVG	LSTM	43.8%	53.5%	2.7%
	SVG	CNN	38.7%	58.2 %	3.1%
	CNN	LSTM	32.7%	66.0%	2.0%
Human 3.6M	SVG	LSTM	34.5%	63.0%	2.5%
	SVG	CNN	96.6%	2.9%	0.4%
	CNN	LSTM	2.5%	97.5%	0.0%
KITTI	SVG	LSTM	55.4%	44.6%	0.0%
	SVG	CNN	97.3%	2.7%	0.0%
	CNN	LSTM	0.7%	99.3%	0.0%

Table C.1: **Amazon Mechanical Turk human worker preference**. We compared the biggest and baseline models from LSTM and SVG'. The bigger models are more frequently preferred by humans.

C.0.5 Device and network details

To scale up the capacity of the model, we use 32 Google TPUv3 Pods (Google, 2018) for each experiment and a batch size of 32. We distribute the training batch such that there is a single batch element in each 16GB TPU. This way we can use each device to the maximum capacity. We first increase K and M together while keeping K to be equals to M. By simply doubling the number of neurons in each layer, we see an improvement. We then continue to increase K and M up to three times the number of neurons in each layer. At this, point we are not able to increase M anymore without running out of memory, and so, we only continue increasing K.

C.0.6 Architecture and hyper-parameters

For the encoder network we use VGG-net (Simonyan and Zisserman, 2015c) up to layer conv3_3 after pooling and a single convolutional layer with output of 128 channels. A mirrored architecture of the encoder is used for the decoder network. For the Convolutional LSTMs used throughout we use a single layer network with 512 units for LSTM$_\psi$ and LSTM$_\phi$, and a two layer network with 512 units for LSTM$_\theta$. Other than that, we follow a similar architecture as Denton and Fergus (2018) including the skip connections from encoder to decoder. We use $\beta = 0.0001$ for all of our experiments. The number of hidden units in z are 64 for the robot arm dataset and 128 for all other datasets.

APPENDIX D

Neural Kinematic Networks for Unsupervised Motion Retargeting

D.1 Quantitative Evaluation per Motion Retargeting Scenario, and Analysis

In this section, we present quantitative evaluation for the different motion retargeting scenarios mentioned in the main text. We then present findings showing how our method significantly outperforms the best performing baseline (copy quaternions and velocities).

In Table D.1, we show results of retargeting motion previously seen during training into two target scenarios: 1) Character has been seen during training, but not performing the motion used as input (left). 2) Character has never been seen during training (right). In Table D.2, we show results of retargeting motion never seen during training into two target scenarios: 1) Character that has been seen during training (left). 2) Character has never been seen during training (right).

From these results, we can observe the benefits of our full adversarial cycle training vs only using cycle training. In both input motion scenarios — seen during training and never

Known Motion / Known Character	
Method	MSE
Ours: Autoencoder Objective	8.61
Ours: Cycle Consistency Objective	5.68
Ours: Adversarial Cycle Consistency Objective	5.35
Ground-truth joint location variance through time: 4.8	

Known Motion / New Character	
Method	MSE
Ours: Autoencoder Objective	2.16
Ours: Cycle Consistency Objective	1.55
Ours: Adversarial Cycle Consistency Objective	1.35
Ground-truth joint location variance through time: 1.5	

Table D.1: Quantitative evaluation of online motion retargeting using mean square error (MSE). Case study: Known motion / known character (left), and known motion / new character (right).

New Motion / Known Character	
Method	MSE
Ours: Autoencoder Objective	6.55
Ours: Cycle Consistency Objective	4.38
Ours: Adversarial Cycle Consistency Objective	4.39
Ground-truth joint location variance through time: 3.6	

New Motion / New Character	
Method	MSE
Ours: Autoencoder Objective	24.16
Ours: Cycle Consistency Objective	23.49
Ours: Adversarial Cycle Consistency Objective	18.02
Ground-truth joint location variance through time: 11.6	

Table D.2: Quantitative evaluation of online motion retargeting using mean square error (MSE). Case study: New motion / known character (left), and new motion / new character (right).

seen during training — retargeting into a never before seen target skeleton results in overall performance improvement. For known input motions, retargeting into a new character results in a performance improvement of 12.9%, while retargeting into a known character results in a performance improvement of 5.8%. Additionally, for new input motions, retargeting into a new character results in a performance improvement of 23.3%, while retargeting into a known character results in a similar performance. In the previous analysis, we can clearly see that learning character behaviors in the training data results in an overall performance boost when the target character has been seen before. Most importantly, learning skeleton conditioned behaviors results in much better generalization to new characters compared to training with cycle alone. However, we can also see that some scenarios reflect larger errors than others. To explain this phenomenon, we measure the movement in the ground-truth motion sequences by computing the average character height normalized joint location variance through time presented at the bottom of each table. This result shows that the more movement there is in the ground-truth sequence, the larger the MSE becomes, thus the larger errors seen on some of the test scenarios previously presented.

Next, we quantitative evaluate our method and the best performing baseline (copy input quaternions and velocities) by separating testing examples into bins based on average movement through time observed in the ground-truth target motion. This evaluation gives us a clearer insight on how much input movement in space each method can handle during retargeting, and how our method is outperforming the best baseline.

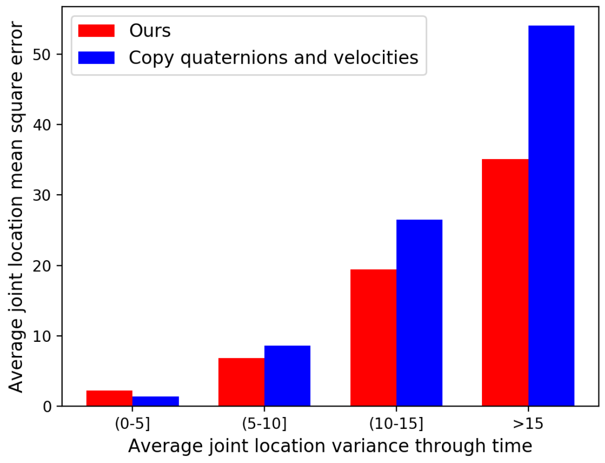


Figure D.1: Quantitative evaluation based on movement through time. The vertical axis denotes mean square error, and the horizontal axis denotes the *xyz*-coordinate average variance through time observed in the ground-truth. The average joint location variance is normalized by character height.

In Figure D.1, we can observe that our method outperforms the baseline as the movement in the evaluation videos increase. Our method substantially outperforms the baseline when the average joint location variance is larger than 5, however, the baseline marginally outperforms our method when the average joint location variance is less than or equals to 5. This result shows that by simply copying the input motion into the target character we cannot guarantee motion retargeting that follows the correct motion in the target character. Therefore, we have to rely on a model that has understanding of the target character and input motion relationships for synthesizing skeleton conditioned motion.

D.2 Denoising 3D Pose Estimation by Motion retargeting

In this section, we show the denoising power of our model on estimated 3D poses. Most 3D pose estimation algorithms do it in a per-frame manner completely ignoring temporal correlations among the estimated poses in videos at every time step. We use our method trained on the Mixamo dataset to retarget the 3D pose estimated by [16] back into the input motion skeleton (Human 3.6M skeleton) to demonstrate denoising effects on the input pose sequence. We compute the Human 3.6M skeleton from the first frame pose in the sequence `WalkTogether 1.60457274` performed by `Subject 9`. We evaluate for denoising by plotting the end-effector height trajectories (hands and feet) of the local motion output of our method since the algorithm we use to estimate 3D pose assumes centered human input. Below we plot end-effector trajectories of our retargeted poses and the originally estimated poses (input to our method) for selected examples (None: Please check our project website for better appreciation of the denoising happening `goo.gl/mDTvem`).

In Figure D.2, we can see that our method denoises the hand end-effectors well, even without having trained with such data before. The feet end-effector denoising is good as well, however in some cases it misses the overall feet height the original estimation had. However, if we take a look at the provided videos, we can clearly see that our method's understanding of the input motion allows it to fix a lot of the shaking seen in the initially estimated 3D pose after motion retargeting.

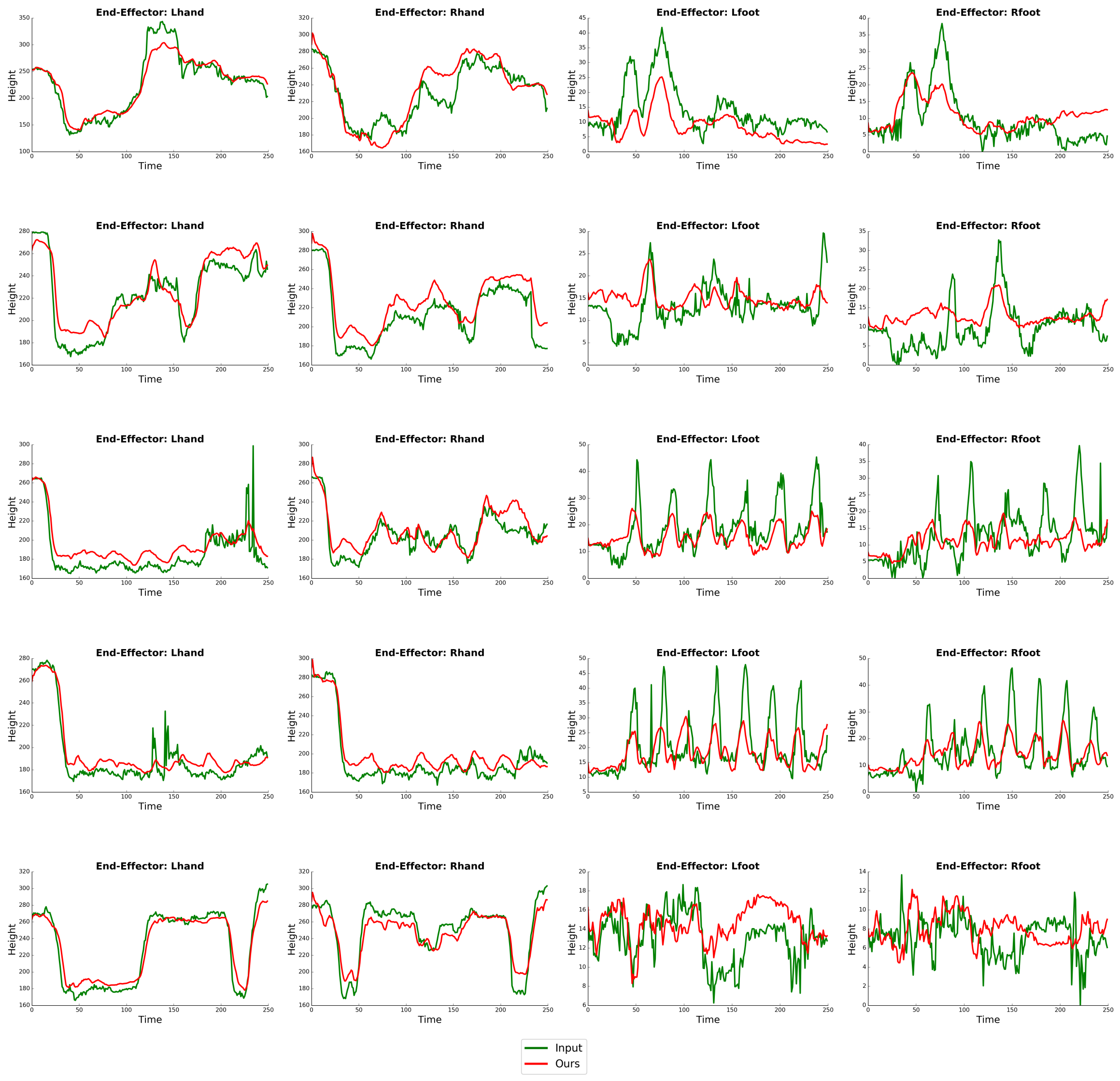


Figure D.2: 3D pose estimation denoising. We present end-effector trajectoriess for 5 examples. Each row belongs a single example in the Human3.6M test set used in [16]. Please refer to our website for visual illustrations of the denoising results. `goo.gl/mDTvem`.

D.3 Data collection process

In this section, we describe the exact steps for collecting the training and testing data from the Mixamo website [1]. As training data, we collected 1656 unique motion sequences distributed over 7 different characters. As testing data, we use 68 unique sequences of at least 4 seconds each (74 total) from which we extract 173 unique non-overlapping 4-second clips (185 total). Please note that the last clip in each sequence which may overlap if there are less that 4 seconds left over after all non-overlapping clips have been extracted. The Mixamo website contains motion separated by pages, the specific pages we downloaded for each character are specified in Table D.3 below:

Training data	
Character	Motion sequence page
Malcolm	[1-5]
Warrok W Kurniawan	[6-10]
Goblin D Shareyko	[11-15]
Kaya	[16-20]
Peasant Man	[21-25]
Big Vegas	[26-30]
AJ	[31-35]

Test data	
Input -> Target	Motion sequence page
Malcolm	28, 51
Warrok W Kurniawan	18, 52
Liam	23, 45
Mutant	33, 45, 52
Claire	52
Sporty Granny	51

Table D.3: Data collection for each character and animation page in the Mixamo website.

At test time, we perform motion retargeting for each testing scenario as shown in Tables D.4 and D.5 below:

Known Motion / Known Character	
Input $\rightarrow$ Target	Motion sequence page
Kaya $\rightarrow$ Warrok W Kurniawan	18
Big Vegas $\rightarrow$ Malcolm	28

Known Motion / New Character	
Input $\rightarrow$ Target	Motion sequence page
Peasant Man $\rightarrow$ Liam	23
AJ $\rightarrow$ Mutant	33

Table D.4: Quantitative evaluation of online motion retargeting using mean square error (MSE). Case study: Known motion / known character (left), and known motion / new character (right).

New Motion / Known Character	
Input $\rightarrow$ Target	Motion sequence page
Sporty Granny $\rightarrow$ Malcolm	51
Claire $\rightarrow$ Warrok W Kurniawan	52

New Motion / New Character	
Input $\rightarrow$ Target	Motion sequence page
Mutant $\rightarrow$ Liam	45
Claire $\rightarrow$ Mutant	52

Table D.5: Quantitative evaluation of online motion retargeting using mean square error (MSE). Case study: New motion / known character (left), and new motion / new character (right).

D.4 Demo video and qualitative evaluation

For the results demo video, please refer to the youtube video link https://youtu.be/BGMyCFmGJWQ (Note: The demo video contains audio. Please wear headphones if you believe you may disturb people around you). For more videos, please go to goo.gl/mDTvem.

D.5 Architecture and training details

In this section, we provide the network architectures detailes used throughout this paper.The RNN architectures are implemented by a 2-layer Gated Recurrent Unit (GRU) with 512-dimensional hidden state. As the discriminator network, we use a 5 layer 1D fully-convolutional neural network with size 4 kernel, and convolutions across. Layers 1-4 have leakyReLU activations with leak of 0.2, dropout of 0.7 keep probability, “same” convolution output with stride 2. Layers 2-4 each have a instance normalization layer with default parameters in the tensorflow implementation. The last layer implements a “valid” convolution with linear activation. For training the networks, we use the Adam optimizer with a learning rate of 1e-4 for both the retargeting RNN and Discriminator, and clip the RNN gradients by global norm of 25. We also implemented a balancing technique between the retargeting network and the discriminator, where the discriminator is not updated if the probability of the generator output being a real falls below 0.3.

BIBLIOGRAPHY

BIBLIOGRAPHY

Adobe's Mixamo (2017). https://www.mixamo.com. Accessed: 2017-09-28.

Alexandre Alahi and Kratarth Goel and Vignesh Ramanathan and Alexandre Robicquet and Li Fei-Fei and Silvio Savarese (2016). Social lstm: Human trajectory prediction in crowded spaces. In *CVPR*.

Ayusawa, K. and Yoshida, E. (2017). Motion retargeting for humanoid robots based on simultaneous morphing parameter identification and motion optimization. *IEEE Trans. on Robotics*.

Babaeizadeh, M., Finn, C., Erhan, D., Campbell, R. H., and Levine, S. (2018). Stochastic variational video prediction. In *ICLR*.

Bagnell, J. A. D. (2015). An invitation to imitation. Technical Report CMU-RI-TR-15-08, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Bin Hammam, G., Wensing, P. M., Dariush, B., and Orin, D. E. (2015). Kinodynamically consistent motion retargeting for humanoids. *IJHR*.

Blender Online Community (2017). *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam.

Brand, M. and Hertzmann, A. (2000). Style machines. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 183–192. ACM Press/Addison-Wesley Publishing Co.

Brock, A., Donahue, J., and Simonyan, K. (2019). Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *ICLR*.

Bütepage, J., Black, M. J., Kragic, D., and Kjellström, H. (2017). Deep representation learning for human motion prediction and classification. In *CVPR*.

Byeon, W., Wang, Q., Srivastava, R. K., and Koumoutsakos, P. (2018). ContextVP: Fully Context-Aware Video Prediction. In *ECCV*.

Chang, M. B., Ullman, T., Torralba, A., and Tenenbaum, J. B. (2017). A compositional object-based approach to learning physical dynamics. In *ICLR*.

Chao, Y.-W., Yang, J., Price, B., Cohen, S., and Deng, J. (2017). Forecasting human dynamics from static images. In *CVPR*.

Choi, K.-J. and Ko, H.-S. (1999). On-line motion retargetting. In *CGA*. IEEE.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*.

Denton, E. and Fergus, R. (2018). Stochastic video generation with a learned prior. In *ICML*.

Denton, E. L. and Birodkar, v. (2017). Unsupervised learning of disentangled representations from video. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4417–4426. Curran Associates, Inc.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. volume abs/1810.04805.

Dosovitskiy, A. and Brox, T. (2016a). Generating images with perceptual similarity metrics based on deep networks. *CoRR*, abs/1602.02644.

Dosovitskiy, A. and Brox, T. (2016b). Inverting Visual Representations with Convolutional Networks. In *CVPR*.

Ebert, F., Finn, C., Dasari, S., Xie, A., and Lee, Alex Levine, S. (2018). Visual Foresight: Model-Based Deep Reinforcement Learning for Vision-Based Robotic Control. volume abs/1812.00568.

Farnebäck, G. (2003). Two-frame motion estimation based on polynomial expansion. *Image analysis*.

Finn, C., Goodfellow, I., and Levine, S. (2016a). Unsupervised learning for physical interaction through video prediction. In *NIPS*.

Finn, C., Goodfellow, I. J., and Levine, S. (2016b). Unsupervised learning for physical interaction through video prediction. In *NeurIPS*.

Fragkiadaki, K., Levine, S., Felsen, P., and Malik, J. (2015a). Recurrent network models for human dynamics. In *ICCV*.

Fragkiadaki, K., Levine, S., Felsen, P., and Malik, J. (2015b). Recurrent network models for human dynamics. In *ICCV*.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. In *IJRR*.

Gleicher, M. (1998). Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014a). Generative adversarial nets. In *NeurIPS*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014b). Generative adversarial nets. In *NIPS*.

Google (2018). Cloud TPUs.

Gorelick, L., Blank, M., Shechtman, E., Irani, M., and Basri, R. (2007). Actions as space-time shapes. *TPAMI*, 29(12):2247–2253.

Goroshin, R., Mathieu, M., and LeCun, Y. (2015). Learning to linearize under uncertainty. In *NeurIPS*.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *ICASSP*, pages 6645–6649. IEEE.

Grochow, K., Martin, S. L., Hertzmann, A., and Popović, Z. (2004). Style-based inverse kinematics. In *ACM transactions on graphics (TOG)*.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein GANs. In *NIPS*.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. (2019). Learning Latent Dynamics for Planning from Pixels. In *ICML*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Heusel, M., Ramsauer, H., Unterthiner, T., and Nessler, B. (2017). GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *NeurIPS*.

Hinton, G. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*.

Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. In *NeurIPS*.

Hoai, M. and Torre, F. (2013). Max-margin early event detectors. *IJCV*.

Holden, D., Saito, J., and Komura, T. (2016). A deep learning framework for character motion synthesis and editing. *TOG*.

Hong, S., Noh, H., and Han, B. (2015). Decoupled deep neural network for semi-supervised semantic segmentation. In *NeurIPS*.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint:1704.04861*.

Hsu, E., Pulli, K., and Popović, J. (2005). Style translation for human motion. In *ACM Transactions on Graphics (TOG)*.

Huang, Y., Cheng, Y., Chen, D., Lee, H., Ngiam, J., Le, Q. V., and Chen, Z. (2018). GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. volume abs/1811.06965.

Ionescu, C., Li, F., and Sminchisescu, C. (2011). Latent structured models for human pose estimation. In *ICCV*.

Ionescu, C., Papava, D., Olaru, V., and Sminchisescu, C. (2014a). Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *TPAMI*.

Ionescu, C., Papava, D., Olaru, V., and Sminchisescu, C. (2014b). Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. *PAMI*.

Jain, A., Zamir, A. R., Savarese, S., and Saxena, A. (2016). Structural-rnn: Deep learning on spatio-temporal graphs. In *CVPR*, pages 5308–5317.

Jayaraman, D. and Grauman, K. (2015). Learning image representations tied to ego-motion. In *ICCV*.

Jayaraman, D. and Grauman, K. (2016). Look-ahead before you leap: end-to-end active recognition by forecasting the effect of motion. *arXiv preprint:1605.00164*.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., Hughes, M., and Dean, J. (2017). Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association of Computational Linguistics*.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Sepassi, R., Tucker, G., and Michalewski, H. (2019). Model-Based Reinforcement Learning for Atari. *CoRR*, abs/1903.00374.

Kalchbrenner, N., Oord, A. v. d., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., and Kavukcuoglu, K. (2017). Video pixel networks. In *ICML*.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *CVPR*.

Kovar, L. and Gleicher, M. (2004). Automated extraction and parameterization of motions in large data sets. In *ACM Transactions on Graphics (ToG)*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NeurIPS*.

Kumar, M., Babaeizadeh, M., Erhan, D., Finn, C., Levine, S., Dinh, L., and Kingma, D. (2018). VideoFlow: A Flow-Based Generative Model for Video. In *ICML*.

Lan, T., Chen, T., and Savarese, S. (2014). A hierarchical representation for future action prediction. In *ECCV*.

Lee, A. X., Zhang, R., Ebert, F., Abbeel, P., Finn, C., and Levine, S. (2018). Stochastic Adversarial Video Prediction. volume abs/1804.01523.

Lee, J. and Shin, S. Y. (1999). A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co.

Lee, N. (2015). *Modeling of Dynamic Environments for Visual Forecasting of American Football Plays*. PhD thesis, Carnegie Mellon University Pittsburgh, PA.

Li, Z., Zhou, Y., Xiao, S., He, C., and Li, H. (2018). Auto-conditioned lstm network for extended complex human motion synthesis. In *ICLR*.

Liang, X., Lee, L., Dai, W., and Xing, E. P. (2017). Dual Motion GAN for Future-Flow Embedded Video Prediction. In *ICCV*.

Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: common objects in context. *ECCV*.

Liu, G., Ceylan, D., Yumer, E., Yang, J., and Lien, J.-M. (2017). Material editing using a physically based rendering network. In *ICCV*. IEEE.

Lotter, W., Kreiman, G., and Cox, D. (2015). Unsupervised learning of visual structure using predictive generative networks. *arXiv preprint arXiv:1504.08023*.

Lotter, W., Kreiman, G., and Cox, D. (2017). Deep predictive coding networks for video prediction and unsupervised learning. In *ICLR*.

Martinez, J., Black, M. J., and Romero, J. (2017a). On human motion prediction using recurrent neural networks. In *CVPR*. IEEE.

Martinez, J., Hossain, R., Romero, J., and Little., J. J. (2017b). A simple yet effective baseline for 3d human pose. *ICCV*.

Mathieu, M., Couprie, C., and LeCun, Y. (2015). Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*.

Mathieu, M., Couprie, C., and LeCun, Y. (2016). Deep multi-scale video prediction beyond mean square error. In *ICLR*.

Mehta, D., Sridhar, S., Sotnychenko, O., Rhodin, H., Shafiei, M., Seidel, H.-P., Xu, W., Casas, D., and Theobalt, C. (2017). Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*.

Merel, J., Tassa, Y., Srinivasan, S., Lemmon, J., Wang, Z., Wayne, G., and Heess, N. (2017). Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*.

Michalski, V., Memisevic, R., and Konda, K. (2014a). Modeling deep temporal dependencies with recurrent "grammar cells". In *NeurIPS*.

Michalski, V., Memisevic, R., and Konda, K. (2014b). Modeling deep temporal dependencies with recurrent “grammar cells”. In *NeurIPS*.

Min, J., Liu, H., and Chai, J. (2010). Synthesis and editing of personalized stylistic human motion. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 39–46. ACM.

Mittelman, R., Kuipers, B., Savarese, S., and Lee, H. (2014). Structured recurrent temporal restricted boltzmann machines. In *ICML*.

Newell, A., Yang, K., and Deng, J. (2016). Stacked hourglass networks for human pose estimation. In *ECCV*.

Oh, J., Guo, X., Lee, H., Lewis, R. L., and Singh, S. (2015). Action-conditional video prediction using deep networks in atari games. In *NeurIPS*.

Patraucean, V., Handa, A., and Cipolla, R. (2015). Spatio-temporal video autoencoder with differentiable memory. *CoRR*, abs/1511.06309.

Pickup, L. C., Pan, Z., Wei, D., Shih, Y., Zhang, C., Zisserman, A., Scholkopf, B., and Freeman, W. T. (2014). Seeing the arrow of time. In *CVPR*.

Pintea, S. L., van Gemert, J. C., and Smeulders, A. W. M. (2014). Dejavu: Motion prediction in static images. In *ECCV*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners. In *Technical report*.

Ranzato, M., Szlam, A., Bruna, J., Mathieu, M., Collobert, R., and Chopra, S. (2014). Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604*.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2018). Regularized evolution for image classifier architecture search. volume abs/1802.01548.

Reed, S., Akata, Z., Mohan, S., Tenka, S., Schiele, B., and Lee, H. (2016a). Learning what and where to draw. In *NeurIPS*.

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., and Lee, H. (2016b). Generative adversarial text-to-image synthesis. In *ICML*.

Reed, S., Zhang, Y., Zhang, Y., and Lee, H. (2015a). Deep visual analogy-making. In *NeurIPS*.

Reed, S. E., Zhang, Y., Zhang, Y., and Lee, H. (2015b). Deep visual analogy-making. In *NIPS*.

Rezende, D. J., Eslami, S. A., Mohamed, S., Battaglia, P., Jaderberg, M., and Heess, N. (2016). Unsupervised learning of 3d structure from images. In *NeurIPS*.

Rose III, C. F., Sloan, P.-P. J., and Cohen, M. F. (2001). Artist-directed inverse-kinematics using radial basis function interpolation. In *Computer Graphics Forum*.

Ryoo, M. S. (2011). Human activity prediction: Early recognition of ongoing activities from streaming videos. In *ICCV*.

Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*.

Schuldt, C., Laptev, I., and Caputo, B. (2004). Recognizing human actions: A local svm approach. In *ICPR*.

Sermanet, P., Lynch, C., Hsu, J., and Levine, S. (2017). Time-contrastive networks: Self-supervised learning from multi-view observation. *arXiv preprint arXiv:1704.06888*.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., and WOO, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NeurIPS*.

Shon, A., Grochow, K., Hertzmann, A., and Rao, R. P. (2006). Learning shared latent structure for image synthesis and robotic imitation. In Weiss, Y., Schölkopf, P. B., and Platt, J. C., editors, *NeurIPS*.

Simonyan, K. and Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In *NeurIPS*.

Simonyan, K. and Zisserman, A. (2015a). Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Simonyan, K. and Zisserman, A. (2015b). Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Simonyan, K. and Zisserman, A. (2015c). Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Soomro, K., Zamir, A. R., and Shah, M. (2012). UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

Srivastava, N., Mansimov, E., and Salakhudinov, R. (2015). Unsupervised learning of video representations using lstms. In *ICML*.

Sutskever, I., Hinton, G. E., and Taylor, G. W. (2009). The recurrent temporal restricted boltzmann machine. In *NeurIPS*.

Sutton, R. (2019). The Bitter Lesson.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *CVPR*.

Tak, S. and Ko, H.-S. (2005). A physically-based motion retargeting filter. *ACM Transactions on Graphics (TOG)*.

Taylor, G. W., Hinton, G. E., and Roweis, S. T. (2007). Modeling human motion using binary latent variables. In *NeurIPS*.

Tulsiani, S., Zhou, T., Efros, A. A., and Malik, J. (2017). Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*.

Tulyakov, S., Liu, M.-Y., Yang, X., and Kautz, J. (2018). Mocogan: Decomposing motion and content for video generation. In *CVPR*.

Unterthiner, T., van Steenkiste, S., Kurach, K., Marinier, R., Michalski, M., and Gelly, S. (2018). Towards Accurate Generative Models of Video: A New Metric & Challenges. *CoRR*, abs/1812.01717.

Villegas, R., Yang, J., Hong, S., Lin, X., and Lee, H. (2017a). Decomposing motion and content for natural video sequence prediction. In *ICLR*.

Villegas, R., Yang, J., Zou, Y., Sohn, S., Lin, X., and Lee, H. (2017b). Learning to generate long-term future via hierarchical prediction. In *ICML*.

Vondrick, C., Pirsiavash, H., and Torralba, A. (2015). Anticipating the future by watching unlabeled video. *arXiv preprint arXiv:1504.08023*.

Vondrick, C., Pirsiavash, H., and Torralba, A. (2016). Generating videos with scene dynamics. In *NeurIPS*.

Walker, J., Doersch, C., Gupta, A., and Hebert, M. (2016). An uncertain future: Forecasting from static images using variational autoencoders. *CoRR*, abs/1606.07873.

Walker, J., Gupta , A., and Hebert , M. (2014). Patch to the future: Unsupervised visual prediction. In *CVPR*.

Wang, J. M., Fleet, D. J., and Hertzmann, A. (2008). Gaussian process dynamical models for human motion. *TPAMI*.

Weinzaepfel, P., Revaud, J., Harchaoui, Z., and Schmid, C. (2013). DeepFlow: Large displacement optical flow with deep matching. In *ICCV*.

Wichers, N., Villegas, R., Erhan, D., and Lee, H. (2018). Hierarchical Long-term Video Prediction without Supervision. In *ICML*.

Wu, J., Lu, E., Kohli, P., Freeman, W. T., and Tenenbaum, J. B. (2017). Learning to see physics via visual de-animation. In *NeurIPS*.

Xia, S., Wang, C., Chai, J., and Hodgins, J. (2015). Realtime style transfer for unlabeled heterogeneous human motion. *ACM Transactions on Graphics (TOG)*.

Xue, T., Wu, J., Bouman, K. L., and Freeman, W. T. (2016). Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. *NeurIPS*.

Yan, X., Rastogi, A., Villegas, R., Sunkavalli, K., Shechtman, E., Hadap, S., Yumer, E., and Lee, H. (2018). Mt-vae: Learning motion transformations to generate multimodal human dynamics. In *ECCV*.

Yan, X., Yang, J., Yumer, E., Guo, Y., and Lee, H. (2016). Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *NeurIPS*.

Yuen, J. and Torralba, A. (2010a). A data-driven approach for event prediction. In *ECCV*.

Yuen, J. and Torralba, A. (2010b). A data-driven approach for event prediction. In *ECCV*.

Yumer, M. E. and Mitra, N. J. (2016). Spectral style transfer for human motion between independent actions. *ACM Transactions on Graphics (TOG)*.

Zeiler, M. D., Taylor, G. W., and Fergus, R. (2011). Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*.

Zhang, W., Zhu, M., and Derpanis, K. G. (2013). From actemes to action: A strongly-supervised representation for detailed action understanding. In *ICCV*, pages 2248–2255.

Zhou, T., Krahenbuhl, P., Aubry, M., Huang, Q., and Efros, A. A. (2016). Learning dense correspondence via 3d-guided cycle consistency. In *CVPR*.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *CVPR*.