LEARNING BEYOND-PIXEL MAPPINGS FROM INTERNET VIDEOS

Yipin Zhou

A dissertation submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Mathematics in the College of Arts and Sciences.

Chapel Hill
2019

Approved by:

Tamara L. Berg

Alexander C. Berg

Marc Niethammer

Jan-Michael Frahm

James Hays

ABSTRACT

Yipin Zhou: Learning beyond-pixel mappings from Internet videos
(Under the direction of Tamara L. Berg)

Recently in the Computer Vision community, there have been significant advancements in algorithms to recognize or localize visual contents for both images and videos, for instance, object recognition and detection tasks. They infer the information that is directly visible within the images or video frames (predicting what's in the frame). While human-level visual understanding could be much more than that, because human also have insights about the information 'beyond the frame'. In other words, people are able to reasonably infer information that is not visible from the current scenes, such as predicting possible future events. We expect the computational models could own the same capabilities one day.

Learning beyond-pixel mappings can be a broad concept. In this dissertation, we carefully define and formulate the problems as specific and subdivided tasks from different aspects. Under this context, what beyond-pixel mapping does is to infer information of broader spatial or temporal context, or even information from other modalities like text or sound. We first present a computational framework to learn the mappings between short event video clips and their intrinsic temporal sequence (which one usually happens first). Then we keep exploring the follow-up direction by directly predicting the future. Specifically we utilize generative models to predict depictions of objects in their future state. Next, we explore a related generation task to generate video frames of the target person with unseen poses guided by a random person. Finally, we propose a framework to learn the mappings between input video frames and it's counterpart in sound domain.

The main contribution of this dissertation lies in exploring beyond-pixel mappings from various directions to add relevant knowledge to the next-generation AI platforms.

To my grandparents

周宜清 and 楊素之

ACKNOWLEDGEMENTS

Throughout my Ph.D., I have received a great amount of support and encouragement from so many people. First, I would like to thank my committee members. I want to express my deepest appreciation to my advisor Tamara Berg. Her professionalism, passion to research and her brilliance greatly motivate and encourage me all along the way. I regard her as a role model for her bravery, kindness and creativity. I can not imagine how my PhD life at UNC could be such happy and wonderful without her and I am so proud and grateful to have her as my advisor. Special thanks to Alex Berg for always being considerate and supportive to the Vision group members. I am impressed and inspired by his deep understanding to the discipline. I also want to thank Jan-Michael Frahm for his insightful feedback and taking care of us besides his own students. Thanks Marc Niethammer for being my committee chair and his continuous support as one of my referrers. Last but not least, I would like to thank James Hays for his teaching and supervision at Brown University as well as enlightening me and leading me into the Computer Vision world.

I also want to acknowledge Dimitris Samaras and Greg Zelinsky from Stony Brook University for the inspired discussion from 3D Vision and psychology perspectives and the cooperation in human gaze tracking project. Special thanks to Nikos Komodakis from Ecole des Ponts ParisTech for his patient supervision and guidance. I spent three wonderful months at Paris and published my very first paper about image deblurring with Nikos. And I would like to acknowledge my amazing mentors from industry for their guidance during the internships. Thanks Yale Song from Yahoo Research, Zhaowen Wang, Chen Fang and Trung Bui from Adobe Research for offering me chance to discuss with and learn from such great researchers. In additions, thanks to Andy van Dam, Pedro Felzenszwalb, Paul Valiant, Enrique Dunn, Jay Aikat, Fabian Monrose and Jan Prins for their teaching, constructing and expanding my knowledge architecture. I would also like to thank staff members at UNC for their great support and help to make my life much easier and smooth, especially to Bridgette Cyr, Jodie Gregoritsch, Melissa Wood, Bil Hays, Mike Stone, John Sopko, Hope Woodhouse and Denise Kenney.

Thanks to all my collaborators, colleagues and labmates: Adam Aji, Akash Bapat, Cheng-Yang Fu, Dinghuang Ji, Dongxu Zhao, Eunbyung Park, Enliang Zheng, Hadi Kiapour, Katharina Schwarz, Hyojin Kim, Jared Heinly, Jie Lei, Johannes Schoenberger, John Lim, Ke Wang, Kiwon Yun, Kota Yamaguchi, Licheng Yu, Marc Eder, Mykhailo (misha) Shvets, Philip Ammirato, Patrick Poirson, Sirion Vittayakorn, Tatiana Tommasi, Thanh Vu, True Price, Vicente Ordonez, Wei Liu, Xufeng Han, Yi Xu, Zhen Wei, Zijun Wei and so many people whose names have not been mentioned. I have learned a lot from each of them and I really cherish the time we spent together. Finally, thanks to my family and friends for their continuous company, unconditional support and love throughout my entire life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AMT Amazon Mechanical Turk. 3, 25, 57, 71

DWT Dynamic Time Warping. 14, 15

FPPA First Person Personalized Activities. 9, 10, 71

GAN Generative Adversarial Network. 3

STN Spatial transformer network. 40

VEGAS Visually Engaged and Grounded AudioSet. 58

CHAPTER 1

Introduction

Computer vision and deep learning technologies affect our daily life from smart supermarkets to autonomous driving. This is in part thanks to the rapid development of algorithms predicting what's in the image/ video frame [Lin et al., 2017a, Lin et al., 2017b, He et al., 2017, Liu et al., 2016, Redmon and Farhadi, 2017, He et al., 2016]. For example, in the Recognition task, we map the input pixels to object labels and in the Semantic segmentation task, the model learns the mapping between images and semantic labels which indicate the object category of each pixel. We human could not only recognize or localize visual contents but also infer the information even if it is not explicitly shown in the current scene. For instance, one is capable of predicting the relationship of a group of people based on their behaviors and the surrounding environments. Or one could make a reasonable guess what might happen next according to the current situation. In this work, the main goal is to train computational models to learn these beyond-pixel mappings and take one step further to the human-level visual understanding. And we narrow down this broad concept into subdivided and specific tasks.

On the other hand, with the emergence of inexpensive video recorders for both third-person and first-person views, cell phones with high-performance cameras as well as video sharing platforms and social media, obtaining self-recorded videos or great number of Internet videos become much easier. The massive amount of the data enable the learning of stronger AI frameworks. Towards this end, we explore making use of the large-scale video data to learn 'beyond-pixel' mappings between visual inputs and broader temporal or spacial context as well as the counterparts from sound domain.

1.1 Related works

There have been several related works exploring temporal perception and future prediction. [Dekel Basha et al., 2012] and [Dekel Basha et al., 2013] recover the temporal ordering of a group of photographs with consistently moving objects using geometric methods. [Pickup et al., 2014a] determines whether an input video is running forwards or backwards based on low-level visual

information. [Yuen and Torralba, 2010, Kitani et al., 2012, Walker et al., 2014, Alahi et al., 2016, Liang et al., 2019] introduce the task of predicting future trajectories of cars, pedestrian or general objects in images or videos. While [Mathieu et al., 2015, Xue et al., 2016, Villegas et al., 2017, Liang et al., 2017, Walker et al., 2017] predict future through pixel-level anticipation, a.k.a generating future video frames/images by utilizing generative models. The above motioned works handle the future frame generation on human poses or general objects within a relatively short period. We explore the future object state prediction in a long range by making use of time-lapse videos.

As discussed by [Xue et al., 2016], predicting future human motions is ambiguous for there can be multiple possibilities, especially for the long-term prediction. Another branch of works formulate the human motion prediction as pose transfer tasks. [Ma et al., 2017, Siarohin et al., 2018, Balakrishnan et al., 2018] generate the depictions a target person in novel poses transferred from a random person. [Neverova et al., 2018] generate images based on an input person image and a surface map (dense 3D pose [Güler et al., 2018]). They present a framework that combines the pixel-level prediction and UV texture mapping. [Liu et al., 2018] presents a method to transfer poses from a source video to the target person. Specifically, they reconstruct a 3D model of the person and train a generative model to produce photo-realistic frames based on images rendered with this 3D model. The work from [Chan et al., 2018] learn the mapping directly from detected poses to generate a target person's video under a personalized setting (train the model only using the data from target person). This works quite well in constrained video domain where models are trained on lab-made videos of a single person showing a large variety of poses. In this thesis, we explore personalized motion transfer on videos obtained from the Internet (the training data can be more flexible). With such uncontrollable settings, our models are required to generalize well to novel poses unseen during the training.

On the other hand, mapping visual information to other non-visual modalities such as text or sound that human could perceive is an important capability for an intelligent agent to communicate with people comprehensively. Work from [Zhang et al., 2017] presents a synthesized audio-visual dataset built by physics/graphics/audio engines. Fine-grained attributes (shape, material and interactions) have been controlled for synthesis. Most related to our work, [Owens et al., 2016] and [Chen et al., 2017] directly generate audio signals conditioned on the input video frames. Specifically, [Owens et al., 2016] predicts hitting sound based on different materials of objects and physical

interactions. A dataset called Greatest Hits (human hit/scratch diverse objects using a drum stick) has been collected for this purpose. [Chen et al., 2017] proposes two generation tasks Sound-to-Image and Image-to-Sound networks using Generative Adversarial Network (GAN) [Goodfellow et al., 2014]. The data used to train the models shows subjects playing various musical instruments indoor with a fixed background. Our work has a similar goal, while instead of mapping visual to sound under constrained/experimental settings, we train neural networks to synthesize raw waveforms and handle more diverse and challenging real-world scenarios.

1.2 Overview

In this section, we briefly introduce the outline of the dissertation. As a prerequisite of future prediction topic, in Chapter 2, we first explore how well human and computers are able to make predictions about temporal ordering of everyday life activities. We introduce two tasks related to temporal ordering prediction. In the first task, given two short video snippets of an activity, the goal is to predict their correct sequence, a.k.a, which snippets usually happen first. The second task is based on the first, given a longer context video plus two video snippets sampled from before or after the context video, the goal is to predict which video snippet was captured closest in time after the context video. This task models the scenario of predicting what a person will do next. These two tasks provide the initial step toward enabling computers to understand the temporal nature of videos. A new dataset of ego-centric videos of everyday activities has been collected through GoPro cameras. This dataset includes both individuals and families living in the same location and allows us to evaluate both general models for temporal prediction and prediction models personalized to a particular individual or environment.

As a natural follow-up, in Chapter 3, we explore predicting the future states. Specifically, we utilize deep generative model to generate the depictions of the future state of objects (we train computational models to learn natural object transformations from videos). We explore several different generation tasks for modeling natural transformations. To enable the learning, we collect time-lapse videos demonstrating four different natural state transformations. Each transformation category includes one or multiple objects. The dataset contains 1463 high-quality time-lapse videos with alignment annotations from Amazon Mechanical Turk (AMT) and has been released publicly. The Figure 1.1 shows the example video frames of time-lapse videos from the dataset.

In Chapter 4, we present a similar generation task with more controllable settings. Specifically,

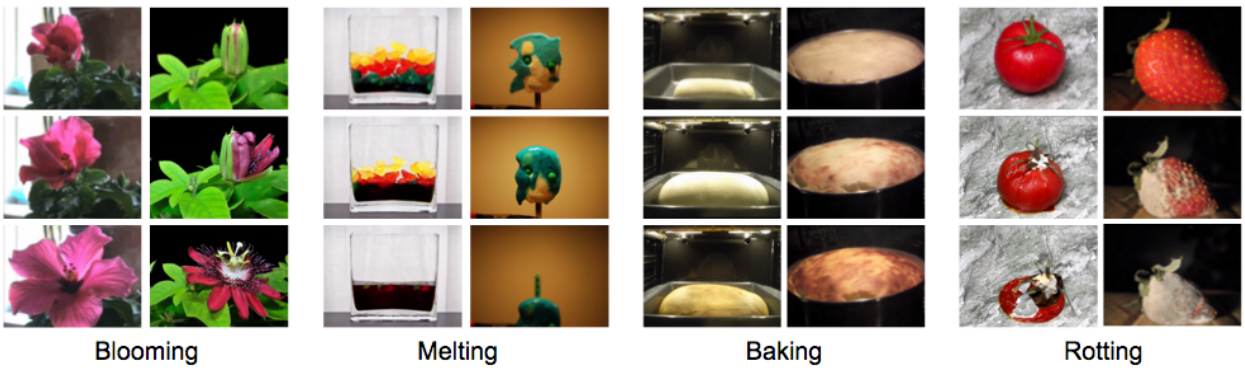


Figure 1.1: Example frames from our dataset of each transformation category: Blooming, Melting, Baking, and Rotting. In each column, time increases as you move down the column, showing how an object transforms.

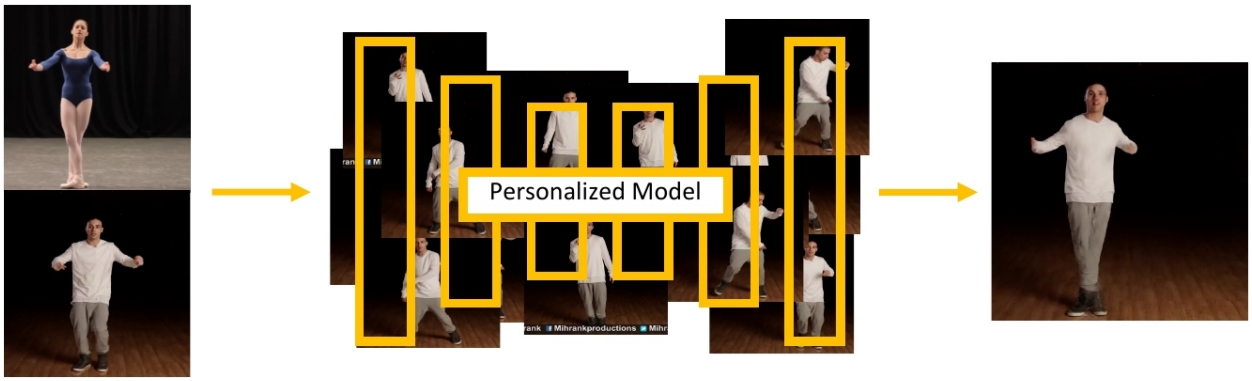


Figure 1.2: We train a personalized model for a target individual in an Internet video. This model can synthesize the target person in novel poses (right) from the input frame (bottom left) driven by a different individual (top left).

we introduce a computational model to transfer any desired movement from a new reference video to the target person. We train this model only using several-minute-long video of the target person performing some random activities as Figure 1.2 shows. Compared with previous or concurrent works which investigate human pose transfer either under generalized settings (models trained across various individuals/scenes) or using single person lab-recorded videos. We we explore personalized motion transfer on videos obtained from the Internet, such as YouTube. Due to the more flexible property of Internet data (backgrounds are non-static or the range of poses demonstrated by the YouTuber are restricted), we break our task down into two sub tasks by proposing a two-stage framework to synthesis foreground human and backgrounds separately.

In Chapter 5, we introduce a framework to map the visual information to raw audio signals. While visual and audio are both important channels to achieve information and they are often

corresponded. The models are expected to learn associations between generated natural sound and visual inputs in the wild for various scenes and object interactions. Due to the missing of benchmark dataset for natural sound generation, we collect a large-scale video-audio dataset with visual and sound modalities closely related (the visual contents/objects are synchronized well with the according sound, e.g. when we hear dog barking sound, there should be dogs shown in the frames) to make the data ideal for the generation tasks. Our dataset is derived from AudioSet [Gemmeke et al., 2017] which is collected for audio event recognition but not ideal for visual to sound mapping task.

The contributes introduced by this dissertation include:

1. Define beyond-pixel mappings concept and formulate the problem through various aspects including temporal perception and future prediction, personalized motion transfer as well as mapping visual information to other modalities.

2. Propose two novel temporal prediction tasks in video domain: video pairwise ordering and future event prediction. Collect a new dataset of ego-centric videos of everyday life activities, including both individuals and families living in the same location.

3. Present a new problem of modeling natural object transformations with deep generative networks by utilizing time-lapse videos. A new dataset of 1463 time-lapse videos depicting 4 common object transformations has been collected.

4. Demonstrate personalized motion transfer on videos from the Internet. Propose a novel two-stage framework to synthesize people performing new movements and fuse them seamlessly with background scenes.

5. Define a new problem of generating sounds (including human/animal and ambient sound) from videos in the wild. Release a dataset designed for generation task containing 28109 cleaned videos (55 hours in total) spanning 10 object categories.

CHAPTER 2

Temporal Sequence Prediction

Given two video fragments of an single event, can we predict which moment usually happens first and next? Understanding the natural of temporal sequence of events is fundamental for future prediction. In this Chapter, we explore predicting temporal context beyond the frame, examining how well humans and computers can make predictions about what usually happens first and what will happen next during everyday activities. Such reasoning will be necessary for producing intelligent agents that can understand what a person is doing and anticipate what they are likely to do next. Both types of inference can help produce robots that more naturally interact with humans in our daily lives as Figure 2.1 shows.

We introduce two tasks related to temporal prediction. In the first task, given two short video snippets of an activity, the goal is to predict their correct temporal ordering. In the second task, given a longer context video plus two video snippets sampled from before or after the context video, the goal is to predict which video snippet was captured closest in time after the context video. These simplified task models the scenario of predicting the temporal sequence and future. The prediction has been applied on Ego-centric video domain (first person video), which has attracted increasing attention due to the rich information contained in egocentric visual data [Fathi et al., 2011a, Pirsiavash and Ramanan, 2012] and easier access to wearable recording devices as well as the potential as as a form of life-blogging [Lu and Grauman, 2013]. We collect a new dataset of ego-centric videos of everyday activities, which allows us to train both generalized and personalized temporal sequence prediction models.

In Section 2.1, we define the two temporal prediction tasks. We describe our first person personalized activity dataset in Section 2.2. Then we apply human experiments to understand the relationship between temporal prediction and the length/interval of the event snippets in Section 2.3. Next we discuss the feature representation and model structions for the learning in Section 2.4 as well as experiments and evaluation results in Section 2.5. This work was described and published in

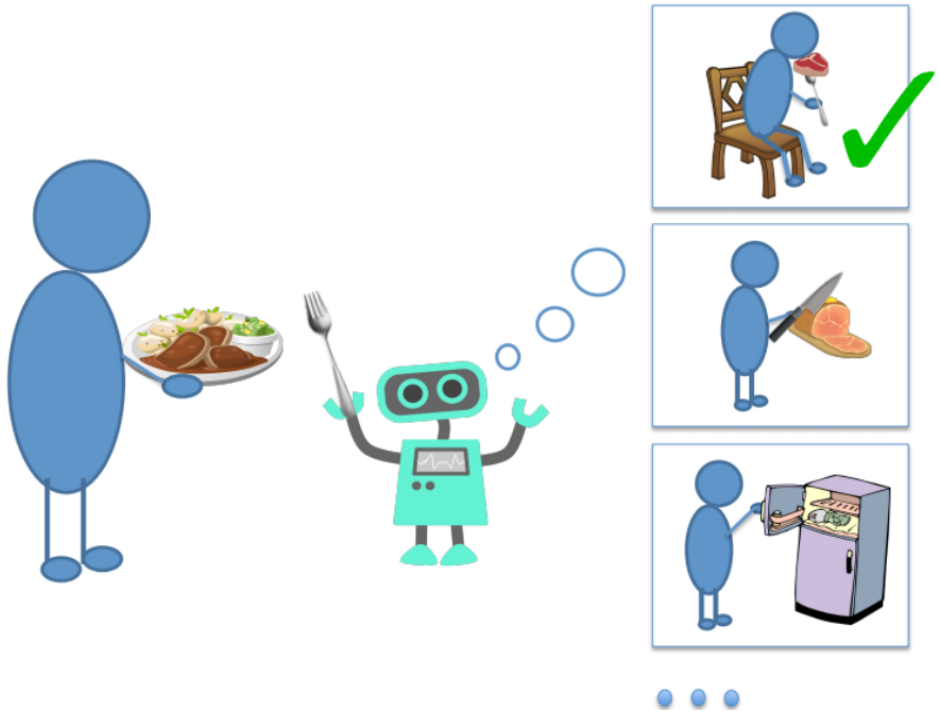


Figure 2.1: If artificial intelligent agents could anticipate what will happen next under the everyday life activity scenarios, they are able to better interact with human.

[Zhou and Berg, 2015].

2.1 Temporal Prediction Tasks

We design two tasks to evaluate temporal understanding. Our goals for designing these tasks are two-fold: 1) we would like to create tasks that are easy to evaluate, and 2) we would like the tasks to be answerable by people so that we can evaluate both human and computer performance. The first task is a pairwise ordering scenario. In this task, we are given two short snippets from an ego-centric video of an activity and asked to infer their correct temporal ordering. For example, in Figure 2.2 the upper box shows two example snippets from the activity of "putting on clothes". Obviously the left snippet occurs before the right snippet in temporal order. Being able to infer the correct temporal ordering between pairs of video snippets is important since it can provide a backbone algorithm for interpreting the temporal information of an entire video sequence or for predicting what might occur next in a video. The second task directly evaluates our ability to make future predictions for everyday activities. In this task, we are provided with a video showing part of an activity plus two shorter video snippets and asked to predict which of the snippets comes next temporally in the video. Figure 2.2 (lower box) shows an example of a person grabbing a bottle and

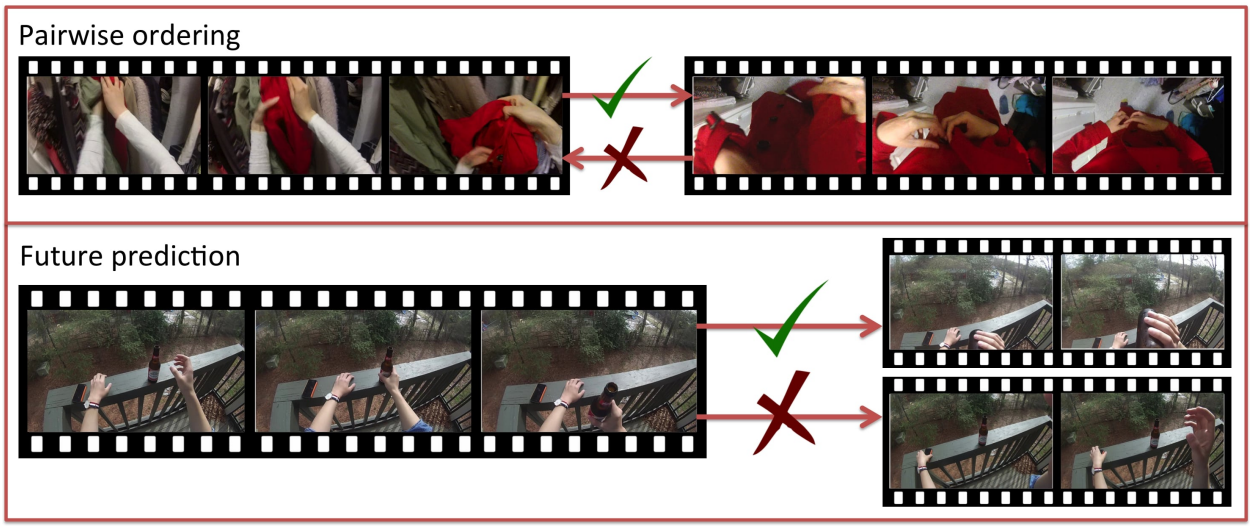


Figure 2.2: Illustration of our two temporal prediction tasks for ego-centric video of everyday activities. In the pairwise ordering task (above) the goal is to provide the correct temporal ordering for two short snippets of video from an activity. In the future prediction task (below), given a longer context video of an activity and two video snippets, the goal is to determine which snippet will occur (closest in time) after the context video.

raising it toward himself. Temporal predictions should tell us that the drinking action is more likely to occur (closest in time) after the context video than the other snippet.

These tasks have several advantages. They provide a measure of video understanding that is complementary to standard activity recognition with tasks that do not require semantic labels, but are easy to evaluate. They also provide quantitative measures for temporal prediction, one challenging aspect of general scene understanding. The tasks are also designed so that we can ask (multiple) people to perform the same tasks as the algorithms, allowing us to measure human performance on temporal prediction. In summary, this provides an initial first step toward enabling computers to understand the temporal nature of videos.

2.2 Dataset

We would like to take advantage of in our models is that we spend much of our daily lives engaged in extremely repetitive activities. Every morning most of us get up, brush our teeth, take a shower, put on clothing, make and eat breakfast, and so on throughout the day. On the other hand, this repetitiveness is offset by the fact that each of us may perform these activities with different variations. For example, one person might prepare cereal for breakfast while someone else may prefer toast. Some people will use a chair to put on their shoes while others may put on their shoes

while standing. All of these factors make for wide variation even in common everyday activities. While one insight is that although these factors, along with variations due to environment, may vary from person to person, a particular person might be quite consistent in how they perform each activity and in the locations in which they perform the activities. Therefore, along with building general temporal prediction models, we would also like to build models for prediction that can be personalized, either to a particular individual or to a particular environment. However, existing ego-centric dataset [Fathi et al., 2012a, Fathi et al., 2012b, Pirsiavash and Ramanan, 2012, Fathi et al., 2011b, Lee et al., 2012] designed for a variety of purposes, such as object recognition, activity recognition, social interaction detection and video summarization contain only one or a few examples of each activity performed by single subject. To enable this personalized learning, we have collected a dataset of first person videos of everyday activities, called the First Person Personalized Activities (FPPA) dataset, where each subject has performed every activity many times. We make use of this dataset for evaluating temporal prediction tasks, but it could also be used for general or personalized activity recognition.

2.2.1 Data collection

For data collection, we make use of a GoPro camera mounted on the user's head using a head strap. The GoPro cameras record ego-centric data simulating the wearer's viewpoint. Each camera captures a high-definition video at 1080p (1920x1080 resolution) with a wide field of view (133.6 degrees) at a rate of 30 frames per second.

We first provide each subject with a list of 5 daily activities (a list of activities is shown in Table 2.1. To encourage subjects to act naturally, they are not provided with any more details. Subjects are encouraged to film themselves completing each activity multiple times at different locations where they would normally perform them. Video recordings were captured in the subjects own homes or in public places (gym, lounge) that they usually visit. In total, our data is made up of 5 sets of videos. Two of the sets consist of videos from a single individual (single-subject) while 3 of the sets consist of videos from families, i.e. two or more people living together at the same location (family subject). We spread the data collection procedure of each set over two months to encourage variability and to gather a large number of videos for each subject.

Activities	Avg videos/subjects	Avg locations/subjects	Total videos/locations
Wash hands	24.2 (19-34)	3.2 (2-7)	121 / 16
Put on shoes	22.8 (21-29)	3.0 (2-6)	114 / 15
Use fridge	26.4 (21-31)	1.6 (1-3)	132 / 8
Drink water	23.2 (16-31)	3.6 (2-7)	116 / 18
Put on clothes	21.6 (16-26)	3.4 (2-5)	108 / 17

Table 2.1: Statistics of FPPA dataset including averaged number of videos from per subject (col2), averaged number of locations per subjects (col 3) and total number of videos and locations (col3). The contents in brackets show the minimal to maximal numbers. The total number of video clips in the dataset is 591.

Figure 2.3: Example frames of 5 activities from our dataset. People perform everyday activities in various locations, and according to their own preferences, e.g. when putting on shoes people might squat down or stand or use a chair.

2.2.2 Characteristics and statistics

The main characteristic of the FPPA dataset is that it is built to enable learning both general and personalized models for temporal prediction. As such we collect a large number of examples of each activity from each subject and for each location. Table 2.1 shows statistics of our dataset, including the per subject average number of the video clips for each activity, and the average number of locations in which each subject performed the activity. On average subjects have performed each activity approximately 20 times. As habits vary from subject to subject, some subjects have performed an activity in a single location while others have performed them in up to 7 different locations. Figure 2.3 shows some example frames from activities performed by different subjects.

2.3 Human experiments

Our main goal is to give computers the ability to predict temporal information for video. Toward this aim, we start with a straightforward pairwise ordering task. However, before we can design even this simple task we would like to know several things: the feasibility of this task for people,

and what specific implementation features should be used for the task, e.g. what length snippets should we use, or how far separated in time should the snippets be so that they are still temporally distinguishable? Therefore, we design two human experiments to gain some useful insights into the pairwise ordering task and then configure our pairwise ordering and future prediction tasks based on these analyses.

2.3.1 Snippet size

We design an experiment to evaluate the effect of snippet length on human perceptions of pairwise ordering using Amazon Mechanical Turk (AMT) as our crowdsourcing platform. For each activity we randomly pick 100 pairs of snippets from videos of the activity, where the central frame for each snippet is selected at a random temporal position within the video. We vary snippet size as 1, 10, 30, 60, or 100 frames (snippet size 1 is a static image). For each pair of snippets we ask 3 AMT workers to tell us which snippet should come first in temporal ordering. To limit bias, we randomly reshuffle the left-right placement of snippets in our Turk interface, and only allow workers to see one snippet length for a particular pair of snippets.

Figure 2.4 (left) shows the effect of snippet size on human performance. One interesting observation is that for the "using fridge" and "drinking water" activities there is an obvious increase in human performance between a snippet size of 1 (static image) and snippet size of 10 frames. One reason for this could be that these activities are relatively symmetric so it may be difficult to tell from a single frame whether the person is opening or closing the fridge, or picking up or putting down a cup. For a video snippet, motion cues help people resolve these ambiguities.

Based on these experiments, we find that human performance for pairwise ordering increases greatly past single frame snippets, but then levels off at about 60 frames. For length 60 frame snippets we find that average performance across users is about 80 to 85% for all activities. Therefore, for the rest of our human and computer experiments we use a snippet length of 60 frames.

2.3.2 Snippet interval

Our next experiment explores how the temporal distance between two snippets affects human pairwise ordering performance. For this experiment, we keep the snippet size fixed to 60 frames, but vary the time interval between the selected snippet pairs. In particular, for each activity we select 100 video snippet pairs. For this sampling, we first select a random video, and a random first snippet. Then we extract the second snippet for the pair from later in the same video with intervals

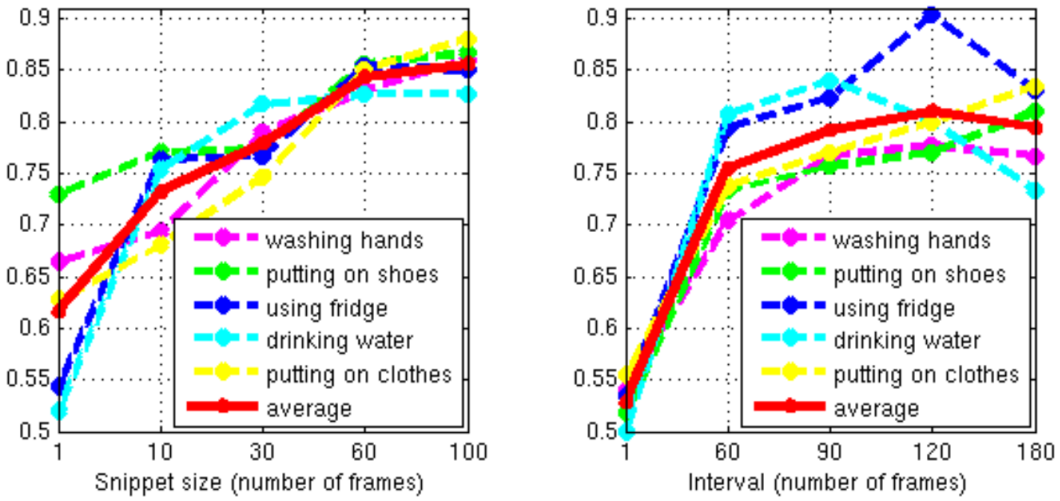


Figure 2.4: Human performance on pairwise ordering. Left shows performance as snippet size varies. Right shows performance as interval varies.

between center frames of 1, 60, 90, 120, or 180 frames. We limit Turker bias in the same manner as the previous experiment.

The results of this experiment are shown in Figure 2.3 (right). Agreeing with intuition, when the temporal offset between two snippets is extremely small (e.g., 1 frame), it is difficult for people to predict the correct pairwise ordering between snippets. As the interval between snippets increases, human performance also increases, but then levels off or even drops as the interval between the snippets gets very long (e.g., 180 frames). For activities such as "putting on clothes" which have clear steps in a relatively lengthy procedure, larger intervals tend to increase performance. For activities like drinking water, the accuracy initially rises with larger intervals then decreases for longer intervals. There are several potential reasons for this: (1) the most distinctive parts of the procedure may be very short, or (2) this activity is periodic (people may repeat the sipping portion of the action multiple times in one drinking activity). Both reasons can cause ambiguity for longer intervals.

2.4 Approaches

2.4.1 Video snippet representation

There has been an increasing amount of work trying to understand the nature of ego-centric video. Different from videos taken from a third person point of view, egocentric videos record not only the inherent environment seen from the wearable device, but also the body or head motion

of the wearer [Aghazadeh et al., 2011]. Quite a few previous works have used techniques such as, hand detection, object detection, or motion features such as optical flow to represent ego-centric videos [Fathi et al., 2011a, Fathi et al., 2012a, Kitani et al., 2011, Pirsiavash and Ramanan, 2012, Ryoo and Matthies, 2013], but how to represent first person videos effectively for temporal prediction tasks is an open question. Recently the use of features learned by deep networks have achieved success in many different vision tasks such as object recognition and detection [Donahue et al., 2013, Girshick et al., 2014], or activity recognition [Simonyan and Zisserman, 2014a]. In this work, we represent video snippets using learned deep features for objects, scenes, and motion.

Object representation: For each frame of a video snippet, we extract the 4096 dimension fc6 layer of the VGG model [Simonyan and Zisserman, 2014b], pre-trained to recognize 1000 ImageNet [Deng et al., 2009] object categories. Then we apply max-pooling over a 10-frame window around the frame to implicitly capture some temporal information about objects within the snippet.

Scene representation: For each frame of a video snippet, to model scene/environment information for video snippets, we extract the 4096 dimension fc7 layer of the Caffe reference model [Jia et al., 2014], pre-trained on the scene-centric Places dataset [Zhou et al., 2014]. Again, we apply max-pooling in a 10-frame window to capture temporal scene information.

Motion representation: Inspired by recent work on deep networks for activity recognition, we re-implement the Temporal Convnet approach of [Simonyan and Zisserman, 2014a]. Their method takes a two stream approach to activity recognition using both object and optical flow features. Our reimplementation of their method achieves an accuracy of 78% on the UCF-101 dataset compared to their reported result of 81%. From the optical flow portion of the Temporal Convnet, we extract the 4096 dimensional fc6 layer as our motion representation.

In our experiments, we evaluate the use of object features in isolation or combining object features with scene or scene and motion features. For the combined features we simply concatenate features for classification. For our experiments we use video snippets of size 60 frames. For each snippet we uniformly sample 6 frames from the snippet, compute the above features and then do max-pooling over each feature dimension to get the final representation.

2.4.2 Pairwise prediction methods

We evaluate several methods for predicting pairwise temporal ordering. The first two are nearest neighbor based methods. The intuition behind these methods is that if two video snippets have similar appearance and motion then we can directly transfer temporal information from one video to the other. One challenge for nearest neighbor methods is that activities can be completed at different rates, therefore we experiment with temporal warping methods to align video snippets. Next we present a regression method that tries to directly predict the time of a video snippet. All three of these models attempt to estimate when a video snippet occurred within a larger activity. Given two snippets we can then predict their pairwise ordering based on their relative estimated times. Our final two methods are trained to directly predict pairwise ordering. Given two video snippets, A and B, we train an SVM and a fully-connected network to predict whether or not snippet A occurred temporally before B. For all of these methods we assume that we know what activity is occurring to focus our efforts on the task of temporal prediction. These methods could be incorporated into a broader system to estimate both activity recognition and temporal predictions.

NN Frac: We first represent the temporal information of all snippets as a real value computed as the temporal position of the snippet relative to the length of the entire action. For each query snippet, using one or more of our feature representations, we retrieve its nearest neighbor snippet from the training set and transfer the nearest neighbor's relative time to the query. Similarity is measured as cosine similarity.

NN DTW: We perform nearest neighbor prediction as before, but a priori first align all training videos for an activity temporally using Dynamic Time Warping (DWT) [Sakoe and Chiba, 1978]. DWT is a dynamic programming algorithm that keeps track of the cost of the best path of the alignment. Here the cost function is defined as the Euclidean distance between each pair of video snippets.

LR: We train a Linear Regression model to estimate the temporal position of a video snippet, relative to the length of the entire activity. Inputs to the regressor are one or more of our feature representations described in Section 2.4.1.

SVM: We train a linear SVM model directly for the pairwise prediction task. Input to the model are concatenated features from a pair of snippets, A and B. Output of the model is a binary prediction $\in \{1, -1\}$ where the model predicts 1 to indicate that A temporally occurs before B and -1 to indicate that A occurs after B. To train this model, we randomly sample pairs of snippets from activities with intervals between the snippets ranging from 60 to 300 frames. The learning parameter is set using cross-validation.

FcNet: Inspired by the metric network architecture introduced in [Han et al., 2015], we train a three layer fully-connected network to predict pairwise ordering. The scenario is the same as the SVM method, that is, we model the task as a binary classification problem. The first two layers of this network have 512 units and use Rectified Linear Unit (ReLU) as the non-linear activation function. The last layer has 2 output units, estimating the probability that snippet A occurs before or after B. During training we apply mini-batch gradient decent and cross-entropy loss. We set the learning rate to be 0.001, dropout rate 0.5, momentum 0.9, and train for 30000 iterations. The hyper-parameters are decided using a validation set.

2.5 Experiments

2.5.1 General pairwise ordering prediction

We evaluate performance of our models on the general pairwise ordering task using a leave-one-out strategy, training on all subjects except one and then predicting on the held out subject. Figure 2.5 (left) shows accuracy averaged over activities and subjects for each prediction method, using a combination of all feature types. Similar to humans, the computational methods do not provide accurate predictions when the interval between snippets is too small, but as the interval increases performance improves. For the NN method applying DWT does not help a great deal, probably due to the high variance of the data. Future work could consider better alignment mechanisms. We also observe that the SVM and FcNet models trained to directly predict temporal ordering outperform the other methods significantly. Table 2.2 shows accuracy for each activity, classifier choice, and feature choice (for interval size 120). For some activities, optimal performance is achieved by combining all of the features while performance in others favored the combination of object and scene features.

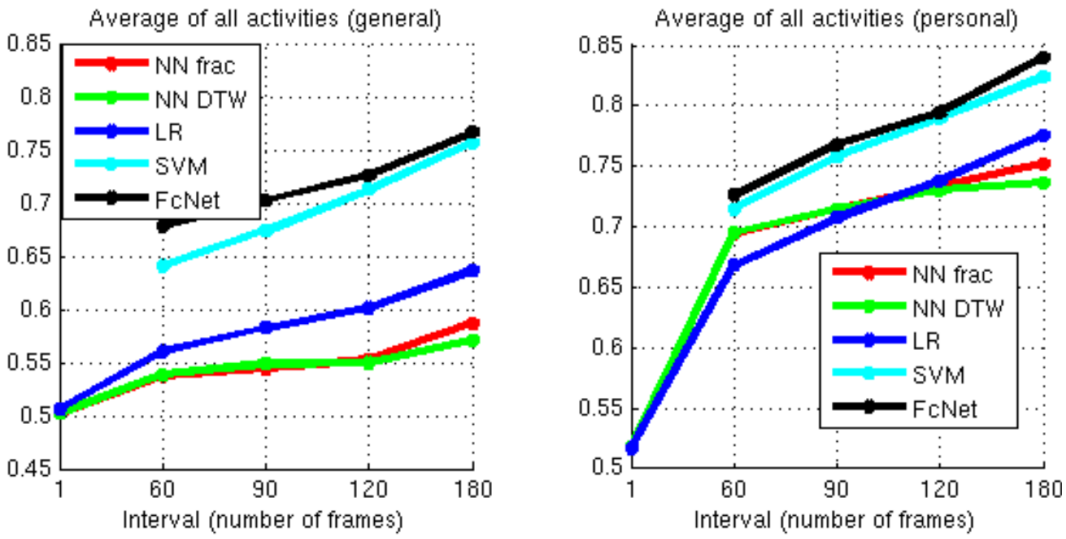


Figure 2.5: Model performance on pairwise ordering. Left shows prediction performance of general models (trained on other subjects). Right shows personalized model performance (trained on other videos from the same subject).

Methods	Washing hands	Putting on shoes	Using fridge	Drinking water	Putting on clothes	Average
NN frac(O)	0.5707	0.5257	0.5494	0.5459	0.5604	0.5504
NN frac(OS)	0.5879	0.5402	0.5468	0.5432	0.5421	0.5521
NN frac(OSM)	0.5694	0.5299	0.5884	0.4938	0.5807	0.5525
NN DTW(O)	0.5835	0.5302	0.5400	0.4709	0.5501	0.5350
NN DTW(OS)	0.5610	0.5447	0.5387	0.5335	0.5533	0.5462
NN DTW(OSM)	0.5738	0.5257	0.5884	0.4855	0.5758	0.5499
LR(O)	0.5635	0.5502	0.6296	0.5844	0.6034	0.5862
LR(OS)	0.5550	0.5853	0.6233	0.5732	0.5923	0.5858
LR(OSM)	0.5536	0.5964	0.6583	0.6039	0.5950	0.6014
SVM(O)	0.6392	0.7026	0.7351	0.6848	0.6967	0.6917
SVM(OS)	0.6609	**0.7270**	**0.7435**	0.6845	**0.7044**	0.7041
SVM(OSM)	**0.6749**	0.6945	0.7402	**0.7538**	0.7006	**0.7128**

Table 2.2: Accuracy of pairwise temporal ordering using general prediction models (trained and tested on different subjects) for interval size 120. Here O indicates object features, S scene features, and M motion features

2.5.2 Personalized pairwise ordering prediction

We evaluate two types of personalized models: models personalized to a particular subject or to a particular location. First, we evaluate personalization where we use different video clips from a single subject for training and testing by applying the leave-one-out method. Since the amount of personalized data is quite limited, for the FcNet, to prevent overfitting, we fine-tune the general network using personalized data for another 25000 iterations (the hyper-parameters remain unchanged). Figure 2.5 (right) shows averaged results across activity, subject, and interval for models personalized to a subject. In these experiments, we see improved performance on the pairwise ordering task compared to general prediction models trained on other subjects, indicating that the personalized models are able to better adapt to a particular individual's habits and daily environments. Table 2.3 shows personalized pairwise ordering accuracy for each activity type and feature combination for interval size of 120. Unlike the general prediction model, the models achieve best performance with the object representation based model, but for some activities incorporating additional features is helpful. We provide several more experiments to further understand personalization. To evaluate generalization for individuals between locations, we train models for the single subjects with different locations in training vs testing. To evaluate generalization between people in the same location, we train models for family-subjects with different family members in training vs. testing. And we also evaluate the effect of training set size. Due to the limited data, for each experiment we apply the SVM method on object, scene, and motion features (Figure 2.6 shows quantitative results). For the individual and location experiments, accuracies are still reasonable compared to the previous personalization experiment. For the training size experiment, we find that the amount of training data required for accurate prediction varies, with some activities benefiting from larger training sizes (e.g. "using fridge") and others achieving surprisingly good accuracy with only 5 samples (e.g. "putting on clothes").

Finally, as mentioned above, pairwise ordering can be used a backbone algorithm for inferring the temporal information of an entire video sequence. To demonstrate this potential, we reshuffle the video sequence of an activity and use our personalized regression model to predict a temporal value for each frame. Then we reorder the frames based on predicted time. Results are visualized in Figure 2.7 where the colorbar shows the reordering of original time (ground truth) information (black=start in original video, white=end), with sampling of reordered keyframes below.

Methods	Washing hands	Putting on shoes	Using fridge	Drinking water	Putting on clothes	Average
NN frac(O)	0.7398	0.7043	0.7491	0.6602	0.7659	0.7239
NN frac(OS)	0.7417	0.7142	0.7626	0.6671	0.7774	0.7326
NN frac(OSM)	0.7304	0.6867	0.7782	0.6934	0.7806	0.7339
NN DTW(O)	0.7547	0.7138	0.7369	0.6616	0.7703	0.7280
NN DTW(OS)	0.7534	0.7010	0.7525	0.6681	0.7784	0.7307
NN DTW(OSM)	0.7288	0.6841	0.7705	0.6942	0.7731	0.7302
LR(O)	0.7118	0.7551	0.8029	0.7233	0.7773	0.7541
LR(OS)	0.6903	0.7425	0.7784	0.6911	0.7567	0.7318
LR(OSM)	0.6872	0.7235	0.8018	0.7248	0.7528	0.7380
SVM(O)	**0.7548**	**0.7822**	0.8136	0.8275	**0.8187**	**0.7994**
SVM(OS)	0.7526	0.7583	0.8208	0.8264	0.8184	0.7953
SVM(OSM)	0.7142	0.7433	**0.8214**	**0.8568**	0.8129	0.7897

Table 2.3: Accuracy of pairwise temporal ordering using personalized prediction models (trained and tested on different videos from the same subject) for interval size 120. Here O indicates object features, S scene features, and M motion features

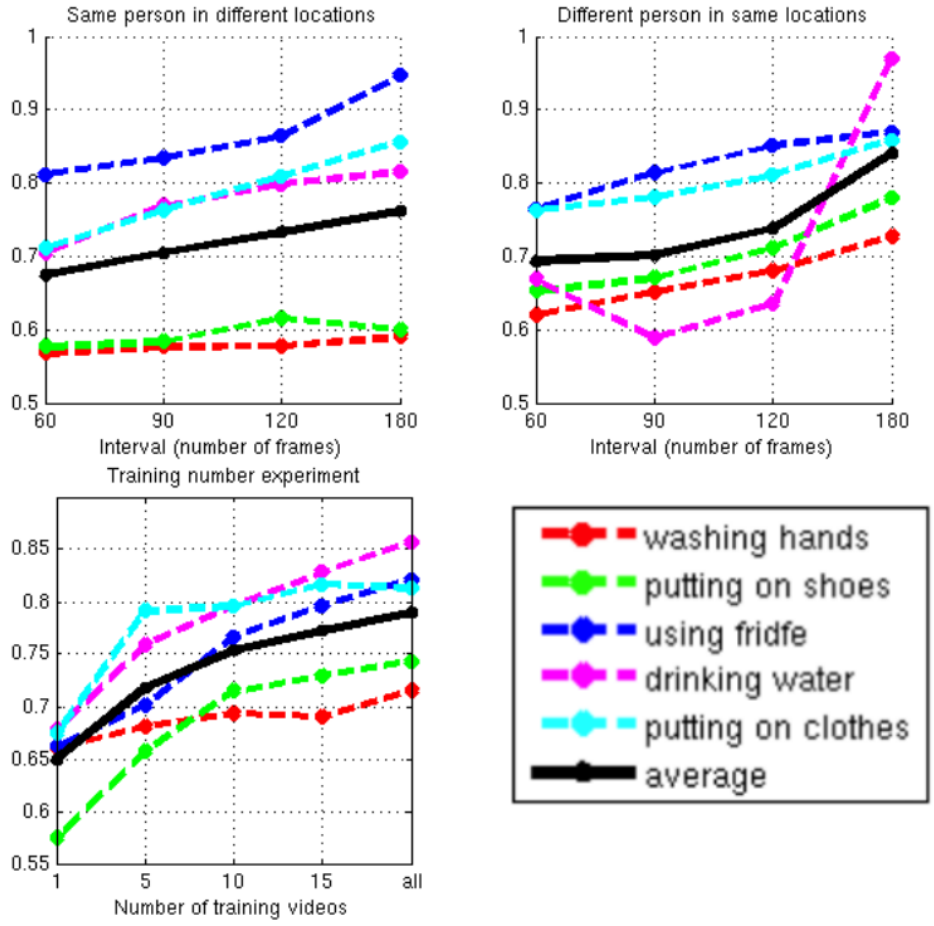


Figure 2.6: Additional personalization experiments for pairwise ordering. Top left shows performance for same person, different locations. Top right shows performance for different people, same location, and bottom left evaluates performance as training set size varies.

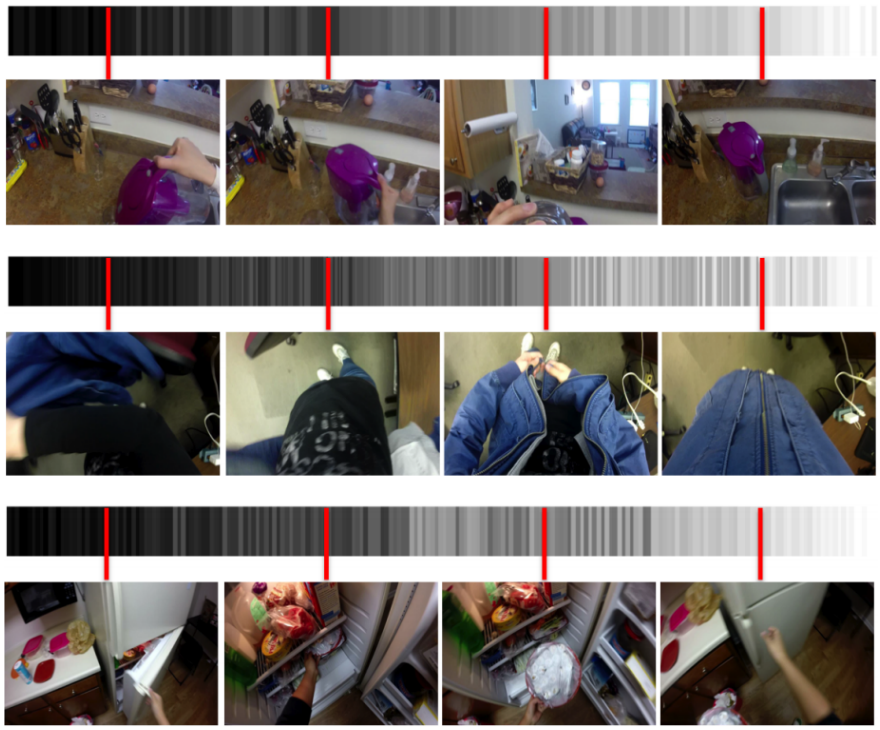


Figure 2.7: Inferring temporal information for an entire video sequence. Colorbar indicates the reordering of original time information (black=start, white=end).

2.5.3 Future prediction task

Next we design the future prediction task where we are provided with a 3 second context video, C, of an activity and two video snippets, A and B, and we are asked to predict which will occur (soonest in time) after C. A is sampled from soon after the context video (randomly selected, but no more than 3 seconds in the future) and B is randomly sampled, either from further in the future or from the time period prior to C. A correct prediction will predict snippet A as happening next.

Computer prediction: Given an algorithm to predict pairwise orderings between snippets, it is straightforward to extend this algorithm to the future prediction task. We compute all pairwise orderings between A, B, and C, and then select the snippet that is most likely to follow after C in temporal order. For these experiments we use combined object, scene, and motion features (Figure 2.8 shows examples).

Human prediction: We also evaluate how well humans can make future predictions when provided with a longer context video. In particular, we show 3 AMT workers the context video plus the

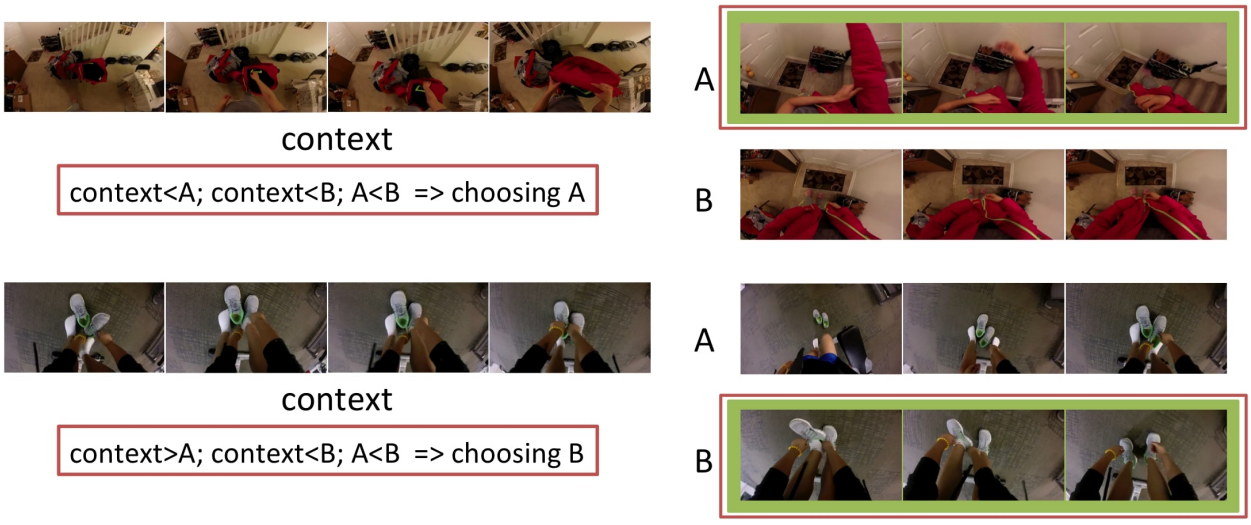
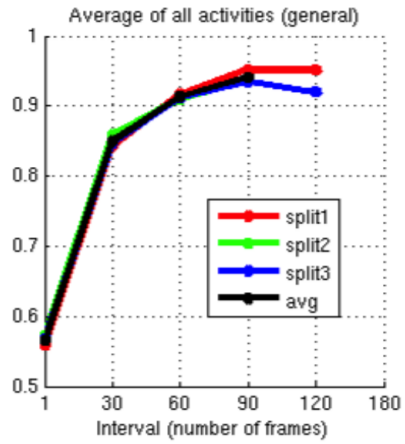


Figure 2.8: Visualization results of computer-based future prediction. Text within red borders are the pairwise ordering results generated by our method. Right shows algorithm proposed future prediction (red border) and ground truth (green border).

Activities	LRg	LRp	SVMg	SVMp	Human
Wash hands	0.5750	0.7050	0.6350	0.7550	0.7816
Put on shoes	0.5700	0.7500	0.7000	0.7250	0.8733
Use fridge	0.6250	0.6800	0.6100	0.7100	0.9284
Drink water	0.5850	0.6700	0.6500	0.7300	0.8717
Put on clothes	0.6350	0.7950	0.7100	0.8350	0.8866
Average	0.5980	0.7200	0.6630	0.7510	0.8686

Table 2.4: Future prediction task accuracy by computational methods and people, where 'g' indicates general model, and 'p' personalized model.



Activities	Our method	FlowWords
wash hands	76.03%	76.86%
put on shoes	69.30%	72.81%
use fridge	79.67%	72.73%
drink water	57.52%	66.09%
put on clothes	80.37%	63.55%
average	72.58%	70.41%

Figure 2.9: Left is the pairwise ordering accuracy of subset of UCF101 dataset. Note that video clips in UCF101 are very short for some actions, the largest interval is only 90. Right is the forward/backward classification accuracy of our method and Flow-Words method in [Pickup et al., 2014b] testing on our dataset

two video snippets and ask the worker to identify which will follow soonest in time after the video. Table 2.4 shows accuracies of human and computer predictions. Personalized models outperform the general models significantly, achieving an average accuracy of 75%. Human performance on this task is also quite good (87%). We also want to understand how well algorithms can perform on the future prediction given snippets from two different videos. Here we first ask humans to perform the future prediction task and then select data with high inter-subject agreement. On this data, using the human predictions as ground truth, the general SVM and FcNet models achieve 66.22% and 66.99% accuracy respectively.

2.5.4 Additional experiments

Pairwise ordering on UCF101: For comparison, we also evaluate our pairwise ordering task on a subset of a wildly used third person action recognition dataset, UCF101. We select 10 categories of action with reasonably long duration and non-repetitive movements. Each category contains more than 100 video clips. In this experiment we evaluate our SVM method and keep other settings (snippet size, feature) the same as our previous experiments. We run 3 train/test splits provided by the UCF official project webpage. Figure 2.9 (left) shows performance. We see that for third person activity videos, our method can achieve even better performance than for first person videos.

Arrow of time: Finally, we use our pairwise ordering method as a backbone to evaluate the task proposed by [Pickup et al., 2014b] on our dataset. The goal is to tell whether a video is runing

forward or backward. We implement their FlowWords classification method which is based on a SIFT-like descriptor and linear SVM. For specific information and parameter settings, please refer to their work. Predicting the temporal direction of video clips can be solved by our pairwise ordering predictions. Our method achieves comparable performance on this task. For each testing video clip, we sample all its snippet pairs with interval 90 and 120 along the video and apply our general SVM model to classify the ordering, then we do majority vote to decide the flow direction of the video. Figure 2.9 (right) shows the average accuracy for Flow-Words and our method.

2.6 Conclusions

We have introduced two tasks for evaluating temporal understanding of ego-centric videos of everyday activities: pairwise ordering and future prediction. We have evaluated both human performance on these tasks and computational models under general and personalized training scenarios. We find that models trained directly on the pairwise ordering task outperform models trained to predict the time at which a video snippet occurred. We also find that personalized models significantly outperform general models, suggesting that to build an accurate predictor for an individual, we should capture data specific to that person.

CHAPTER 3

Future object state prediction

Based on life-long observations of physical, chemical, and biologic phenomena in the natural world, humans can often easily picture in their minds what an object will look like in the future. But, what about computers? In this paper we seek to learn computational models of how the world works through observation. In particular, we explore the use of generative models to create depictions of objects at future times. To enable this learning, we collect time-lapse videos demonstrating four different natural state transformations: melting, blooming, baking, and rotting. Several of these transformations are applicable to a variety of objects. We train models for each transformation - irrespective of the object undergoing the transformation - under the assumption that these transformations have shared underlying physical properties that can be learned. Future prediction has been explored in previous works for generating the next frame or next few frames of a video [Srivastava et al., 2015, Mathieu et al., 2015, Ranzato et al., 2014]. Our focus, in comparison, is to model general natural object transformations and to model future prediction at a longer time scale (minutes to days) than previous approaches.

We explore several different generation tasks for modeling natural transformations. The first task is to generate the future state depiction of an object from a single image of the object. Here the input is a frame depicting an object at time t, and output is a generated depiction of the object at time t+k, where k is specified as a conditional label input to the network. For this task we explore two auto-encoder based architectures. The first architecture is a baseline algorithm built from a standard auto-encoder framework. The second architecture is a generative adversarial network where in addition to the baseline auto-encoder, a discriminator network is added to encourage more realistic outputs. For our second and third future prediction tasks, we introduce different ways of encoding time in the generation process. We propose a two-stack model (second task) that the input is two images of an object at time t and t+m and the model learns to generate a future image according to the implicit time gap between the input images (ie generate a prediction of the object at time

Rotting: 185	Melting: 453	Baking: 242	Blooming: 583
Strawberry: 35	Ice cream: 128	Cookies: 55	Flower: 583
Watermelon: 9	Chocolate: 18	Bread: 57	
Tomato: 25	Butter: 9	Pizza: 59	
Banana: 26	Snow: 54	Cake: 48	
Apple: 23	Wax: 60	Other: 23	
Peach: 8	Ice: 184		
Other: 59			

Table 3.1: Statistics of our transformation categories. Some categories contain multiple objects (e.g. ice cream, chocolate, etc melting) while others apply only to a specific object (e.g. flowers blooming). Values indicate the total number of videos collected for each category.

t+2m). These models are trained on images with varying time gaps. In our last prediction task, our goal is to recursively generate the future states of an object given a single input image of the object. For this task, we use a recurrent neural network to recursively generate future depictions. For each of the described future generation tasks, we also explore the effectiveness of pre-training on a large set of images, followed by fine-tuning on time-lapse data for improving performance.

We first describe the time-lapse dataset and the data collection in Section 3.1. In Section 3.2, we discuss three future generation tasks we mentioned above and the training approaches. Finally, we provide both qualitative and quantitative evaluation results as well as the human experiments in Section 3.3.

3.1 Time-lapse video dataset

Given the high-level goal of understanding temporal transformations of objects, we require a collection of videos showing temporal state changes. Therefore, we collect a large set of object-centric timelapse videos from the web. Timelapse videos are ideal for our purposes since they are designed to show an entire transformation (or a portion of a transformation) within a short period of time.

3.1.1 Data collection

We observe that time-lapse photography is quite popular due to the relative ease of data collection. A search on YouTube for "time lapse" results in over 11 million results. Anyone with a personal camera, GoPro, or even a cell phone can capture a simple time-lapse video and many people post and publicly share these videos on the web. We collect our object-based time-lapse video dataset by directly querying keywords through the YouTube API. For this paper, we query 4 state transformation categories: Blooming, Melting, Baking and Rotting, combined with various object categories. This

results in a dataset of more than 5000 videos. This dataset could be extended to a wider variety of transformations or to more complex multi-object transformations, but as a first step we focus on these 4 as our initial goal set of transformations for learning.

For ease of learning, ideally these videos should be object-centric with a static camera capturing an entire object transformation. However, many videos in the initial data collection stage do not meet these requirements. Therefore, we clean the data using AMT as a crowdsourcing platform. Each video is examined by 3 Turkers who are asked questions related to video quality. Videos that are not timelapse, contain severe camera motion or are not consistent with the query labels are removed. We also manually adjust parts of the videos which are playing backwards (a technique used in some time-lapse videos), contain more than one round of the specified transformation, and remove irrelevant prolog/epilog. Finally our resulting dataset contains 1463 high quality object based timelapse videos. Table 3.1 shows the statistics of transformation categories and their respective object counts. Figure 1.1 (Section 1.2) shows example frames of each transformation category.

3.1.2 Transformation degree annotation

To learn natural transformation models of objects from videos, we need to first label the degree of transformation throughout the videos. In language, people use text labels to describe different states, for instance, fresh vs rotted apple. However, the transformation from one state to another is really a continuous evolution. Therefore, we represent the degree of transformation with a real number, assigning the start state a value of 0 (not at all rotten) and the end state (completely rotten) a value of 1. To annotate objects from different videos we could naively assign the first frame a value of 0 and the last frame a value of 1, interpolating in between. However, we observe that some time-lapse videos may not depict entire transformations, resulting in poor alignments.

Therefore, we design a labeling task for people to assign degrees of transformation to videos. Directly estimating a real value for frames turns out to be impractical as people may have different conceptions of transformation degree. Instead our interface displays reference frames of an object category-transformation and asks Turkers to align frames from a target video to the reference frames according to degree of transformation. Specifically, for each object category transformation pair we select 5 reference frames from a reference video showing: transformation degree values of 0, 0.25, 0.5, 0.75, and 1. Then, for the rest of the videos in that object-transformation category, we ask Turkers to select frames visually displaying the same degree of transformation as the reference frames. If the

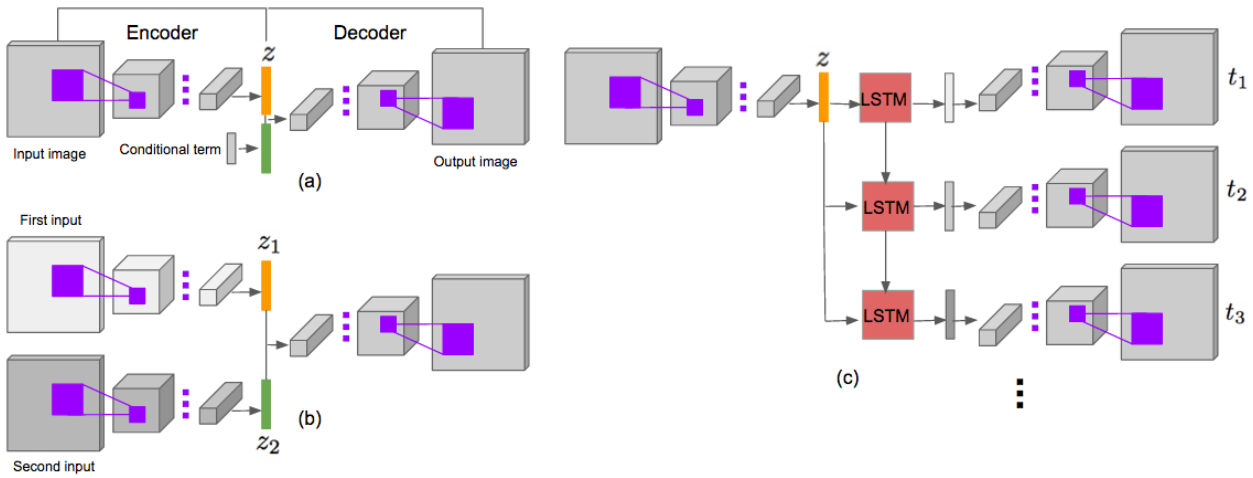


Figure 3.1: Model architectures of three generation tasks: (a) Pairwise generator; (b) Two stack generator; (c) Recurrent generator.

displayed video does not depict an entire transformation, Turkers may align less than 5 frames with the reference. Each target video is aligned by 3 Turkers and the median of their responses is used as annotation labels(linearly interpolating degrees between labeled target frames). This provides us with consistent annotations of transformation degree between videos.

3.2 Future state generation

Our goal is to generate depictions of the future state of objects. In this work, we explore frameworks for 3 temporal prediction tasks. In the first task (Section 3.2.1), called pairwise generation, we input an object-centric frame and the model generates an image showing the future state of this object. Here the degree of the future state transformation - how far in the future we want the depiction to show - is controlled by a conditional term in the model. In the second task (Section 3.2.2) we have two inputs to the model: two frames from the same video showing a state transformation at two points in time. The goal of this model is to generate a third image that continues the trend of the transformation depicted by the first and second input. We call this "two stack" generation. In the third task (Section 3.2.3), called "recurrent generation", the input is a single frame and the goal is to recursively generate images that exhibit future degrees of the transformation in a recurrent model framework.

3.2.1 Pairwise generation

In this task, we input a frame and generate an image showing the future state. We model the task using an autoencoder style convolutional neural network. The model architecture is shown in

Figure 3.1(a), where the input and output size are 64x64 with 3 channels and encoding and decoding parts are symmetric. Both encoding and decoding parts consists of 4 convolution/deconvolution layers with the kernel size 5x5 and a stride of 2, meaning that at each layer the height and width of the output feature map decreases/increases by a factor of 2 with 64, 128, 256, 512/512, 256, 128, 64 channels respectively. Each conv/deconv layer, except the last layer, is followed by a batch normalization operation [Ioffe and Szegedy, 2015] and ReLU activation function [Nair and Hinton, 2010]. For the last layer we use Tanh as the activation function. The size of the hidden variable z (center) is 512. We represent the conditional term encoding the degree of elapsed time between the input and output as a 4 dimensional one-hot vector, representing 4 possible degrees. This is connected with a linear layer (512) and concatenated with z to adjust the degree of the future depiction. Below we describe experiments with different loss functions and training approaches for this network.

p_mse: As a baseline, we use pixel-wised mean square error(p_mse) between prediction output and ground truth as the loss function. Previous image generation works [Srivastava et al., 2015, Mathieu et al., 2015, Ranzato et al., 2014, Pathak et al., 2016] postulate that pixel-wised l_2 criterion is one cause of output generation blurriness. We also observe this effect in the outputs produced by this baseline model.

p_mse+adv: Following the success of recent image generation works[Goodfellow et al., 2014, Radford et al., 2015], which make use of generative adversarial networks (GANs) to generate better quality images, we explore the use of this network architecture for temporal generation. For their original purpose, these networks generate images from randomly sampled noise. Here, we use a GAN to generate future images from an image of an object at the current time. Specifically, we use the baseline autoencoder previously described and incorporate an adversarial loss in combination with the pixel-wise MSE. During training this means that in addition to the auto-encoder, called the generator (G), we also train a binary CNN classifier, called the discriminator (D). The discriminator takes as input, the output of the generator and is trained to classify images as real or fake (i.e. generated). These two networks are adversaries because G is trying to generate images that can fool D into thinking they are real, and D is trying to determine if the images generated by G are real. Adding D helps to train a better generator G that produces more realistic future state images. The

architecture of D is the same as the encoder of G, except that the last output is a scalar and connect with sigmoid function. D also incorporates the conditional term by connecting a one-hot vector with a linear layer and reshaping to the same size as the input image then concatenating in the third dimension. The output of D is a probability, which will be large if the input is a real image and small if the input is a generated image. For this framework, the loss is formulated as a combination of the MSE and adversarial loss:

$$L_G = L_{p\_mse} + \lambda_{adv} * L_{adv} \tag{3.1}$$

where L_{p_mse} is the mean square error loss, where x is the input image, c is the conditional term, G(.) is the output of the generation model, and y is the ground truth future image at time = current time + c.

$$L_{p\_mse} = |y - G(x, c)|^2 \tag{3.2}$$

And, L_{adv} is a binary cross-entropy loss with $\alpha = 1$ that penalizes if the generated image does not look like a real image. D[.] is the scalar probability output by the discriminator.

$$L_{adv} = -\alpha \log(D[G(x, c), c]) - (1 - \alpha) \log(1 - D[G(x, c), c]) \tag{3.3}$$

During training, the binary cross-entropy loss is used to train D on both real and generated images. For details of jointly training adversarial structures, please refer to [Goodfellow et al., 2014].

p_g_mse+adv: Inspired by [Mathieu et al., 2015], where they introduce gradient based l_1 or l_2 loss to sharpen generated images. We also evaluate a loss function that is a combination of pixel-wise MSE, gradient based MSE, and adversarial loss:

$$L_G = L_{p\_mse} + L_{g\_mse} + \lambda_{adv} * L_{adv} \tag{3.4}$$

where L_{g_mse} represents mean square error in the gradient domain defined as:

$$L_{g\_mse} = (g_x[y] - g_x[G(x, c)])^2 + (g_y[y] - g_y[G(x, c)])^2 \tag{3.5}$$

$g_x[.]$ and $g_y[.]$ are the gradient operations along the x and y axis of images. We could apply different weights to L_{p_mse} and L_{g_mse}, but in this work we simply weight them equally.

p_g_mse+adv+ft: Since we have limited training data for each transformation, we investigate the use of pre-training for improving generation quality. In this method we use the same loss function as the last method, but instead of training the adversarial network from scratch, training proceeds in two stages. The first stage is a reconstruction stage where we train the generation model using random static images for the reconstruction task. Here the goal is for the output image to match the input image as well as possible even though passing through a bottleneck during generation. In the second stage, the fine-tuning stage, we fine-tune the network for the temporal generation task using our timelapse data. By first pre-training for the reconstruction task, we expect the network to obtain a good initialization, capturing a representation that will be useful for kick-starting the temporal generation fine-tuning.

3.2.2 Two stack generation

In this scenario, we want to generate an image that shows the future state of an object given two input images showing the object in two stages of its transformation. The output should continue the transformation pattern demonstrated in the input images, i.e. if the input images depict the object at time t and t+m, then the output should depict the object at time t+2m. We design the generation model using two stacks in the encoding part of the model as shown in Fig. 3.1(b). The structures of the two stacks are the same and are also identical to the encoding part of the pairwise generation model. The hidden variables z_1 and z_2 are both 512 dimensions, and are concatenated together and fed into the decoding part, which is also the same as the previous pairwise generation model. The two stacks are sequential, trained independently without shared weights.

Given the blurry results of the baseline for pairwise generation, here we only use three methods **p_mse+adv**, **p_g_mse+adv**, and **p_g_mse+adv+ft**. The structure of the discriminator is the same. For the fine-tuning method, during the reconstruction training, we make the two inputs the same static image. The optimization procedures are the same as for the pairwise generation task, but we do not have conditional term here (since time for the future generation is implicitly specified by the input images).

3.2.3 Recurrent generation

In this scenario, we would like to recursively generate future images of an object given only a single image of its current temporal state. In particular, we use a recurrent neural network framework for generation where each time step generates an image of the object at a fixed degree interval in the future. This model structure is shown in Fig. 3.1(c). After hidden variable z, we add a LSTM[Hochreiter and Schmidhuber, 1997] layer. For each time step, the LSTM layer takes both z and the output from the previous time step as inputs and sends a 512 dimension vector to the decoder. The structure of the encoder and decoder are the same as in the previous scenarios, where the decoding portions for each time slot share the same weights.

We evaluate three loss functions in this network: **p_mse+adv**, **p_g_mse+adv** as well as **p_g_mse+adv+ft**. The structure of the discriminator is again the same without the conditional term (as in the two-stack model). For fine-tuning, during reconstruction training, we train the model to recurrently output the same static image as the input at each time step.

3.3 Experiments

In this section, we discuss the training process and parameter settings for all experiments (Sec 3.3.1). Then, we describe dataset pre-processing and augmentation (Sec 3.3.2). Finally, we discuss quantitative and qualitative analysis of results for: pairwise generation (Sec 3.3.3), two-stack generation (Sec 3.3.4), and recurrent generation (Sec 3.3.5) as well as the human experiments and the visualization results (Sec 3.3.6).

3.3.1 Training and parameter settings

Unless otherwise specified training and parameter setting details are applied to all models. During training, we apply Adam Stochastic Optimization[Kingma and Ba, 2014] with learning rate 0.0002 and minibatch of size 64. In the loss functions where we combine mean square error loss with adversarial loss (as in equation 3.1), we set the weight of the adversarial loss to $\lambda_{adv} = 0.2$ for all experiments.

3.3.2 Dataset preprocessing and augmentation

Timelapse dataset: Some of the collected videos depict more than one object or the object is not located in the center of the frames. In order to help the model concentrate on learning the transformation itself rather object localization, for each video in the dataset, we obtain a cropped version of the frames centered on the main object. We randomly split the videos into training and

testing sets in a ratio of 0.85 : 0.15. Then, we sample frame pairs (for pairwise generation) or groups of frames (for two-stack and recurrent generation) from the training and testing videos. Frames are resized to 64x64 for generation. To prevent overfitting, we also perform data augmentation training pairs or groups by incorporating frame crops and left-right flipping.

Reconstruction dataset: This dataset contains static images used to pre-train our models on reconstruction tasks. Initially, we tried training only on objects depicted in the timelapse videos and observed performance improvement. However, collecting images of specific object categories is a tedious task at large-scale. Therefore, we also tried pre-training on random images (scene images or images of random objects) and found that the results were competitive. This implies that the content of the images is not as important as encouraging the networks to learn how to reconstruct arbitrary input images well. We randomly download 50101 images from ImageNet[Deng et al., 2009] as our reconstruction dataset. The advantage of this strategy is that we are able to use the same group of images for every transformation model and task.

3.3.3 Pairwise generation results

In the pairwise generation task, we input an image of an object and the model outputs depiction of the future state of the object, where the degree of transformation is controlled by a conditional term. The conditional term is a 4 dimensional one-hot vector which indicates whether the predicted output should be 0, 0.25, 0.5 or 0.75 degrees in the future from the input frame. We sample frame pairs from timelapse videos based on annotated degree value intervals. A 0 degree interval means that the input and output training images are identical. We consider 0 degree pairs as auxiliary pairs, useful for two reasons: 1) They help augment the training data with video frames (which display different properties from still images), and 2) The prediction quality of image reconstruction is highly correlated with the quality of future generation. Pairs from 0 degree transformations can be easily be evaluated in terms of reconstruction quality since the prediction should ideally exactly match the ground truth image. Predictions for future degree transformations are somewhat more subjective. For example, from a bud the resulting generated bloom may look like a perfectly valid flower, but may not match the exact flower shape that this particular bud grew into (an example is shown in Figure. 3.2 row 1).

We train pairwise generation models separately for each of the 4 transformation categories using

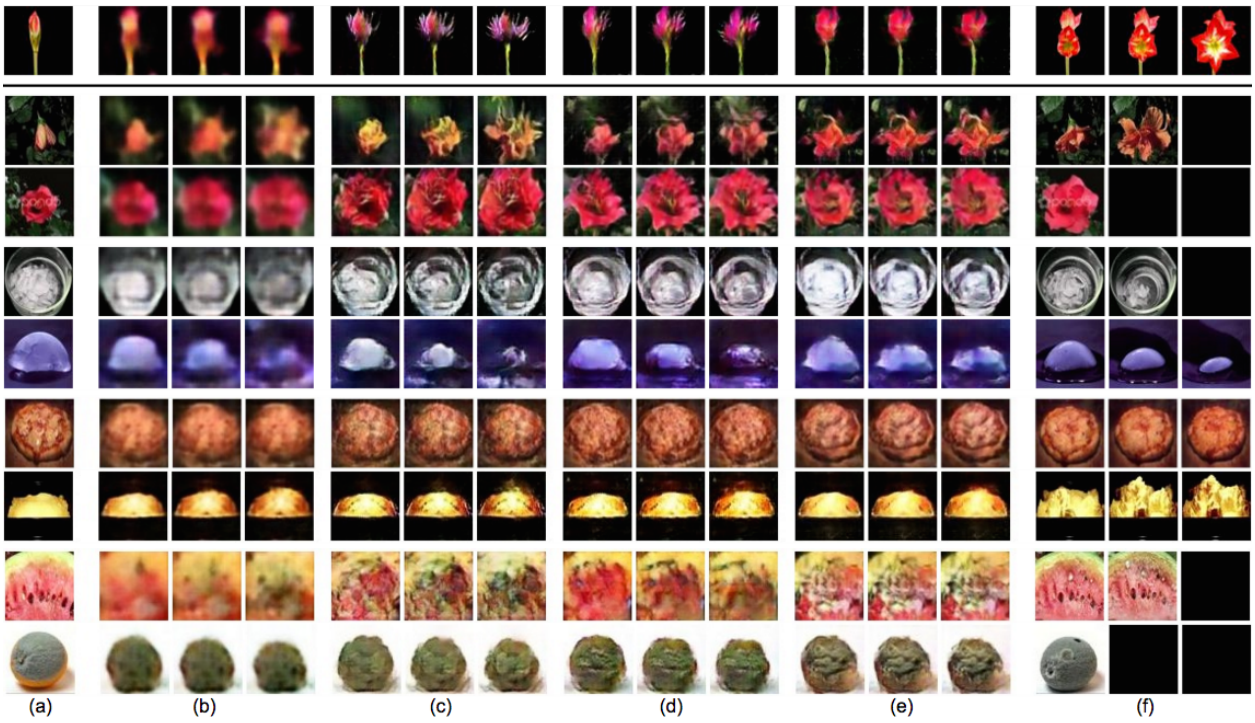


Figure 3.2: Pairwise generation results for: Blooming (rows 1-3), Melting (rows 4-5), Baking (rows 6-7) and Rotting (rows 8-9). Input(a), p_mse(b), p_mse+adv(c), p_g_mse+adv(d), p_g_mse+adv+ft(e), Ground truth frames(f). Black frames in the ground truth indicate video did not depict transformation that far in the future.

p_mse, **p_mse+adv** and **p_g_mse+adv** methods, trained for 12500 iterations on timelapse data from scratch. For the **p_g_mse+adv+ft** method, the models are first trained on the reconstruction dataset for 5000 iterations with the conditional term fixed as '0 degree' and then fine-tuned on timelapse data for another 5500 iterations. We observe that the fine-tuning training converges faster than training from scratch (example results with 3 different degree conditional terms in Figure 3.2). We observe that the baseline suffers from a high degree of blurriness. Incorporating other terms into the loss function improves results, as does pre-training with fine-tuning. Table. 3.2 (cols 2-4) shows evaluations of pairwise-generation reconstruction. For evaluation, we compute the Peak Signal to Noise Ratio(PSNR), Structural Similarity Index(SSIM) and Mean Square Error (MSE) values between the output and ground truth. We can see that incorporating gradient loss slightly improves results while pre-training further improves performance. This agrees with the qualitative visual results.

	Pairwise			Two stack			Recurrent		
	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE
p_mse+adv	17.0409	0.5576	0.0227	17.7425	0.5970	0.0185	17.2758	0.5747	0.0211
p_g_mse+adv	17.0660	0.5720	0.0224	17.9157	0.6122	0.0177	17.2951	0.5749	0.0214
p_g_mse+adv+ft	17.4784	0.6036	0.0207	18.6812	0.6566	0.0153	18.3357	0.6283	0.0166

Table 3.2: Quantitative evaluation of: Pairwise generation, Two stack generation, and Recurrent tasks. For PSNR and SSIM larger is better, while for MSE lower is better.

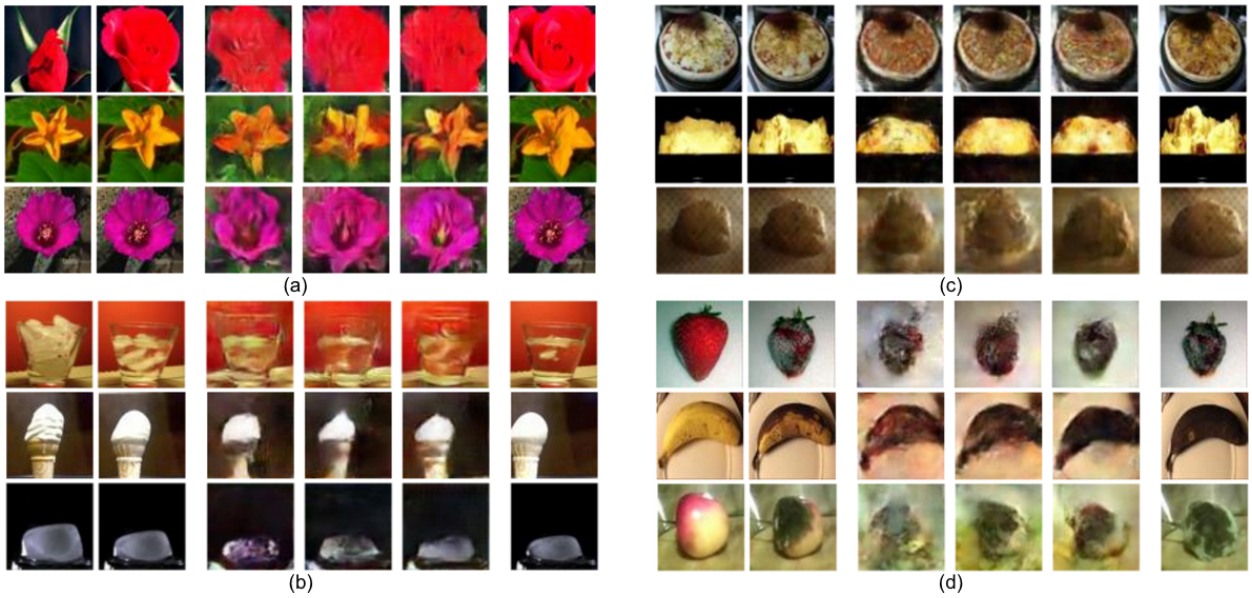


Figure 3.3: Two stack generation results for Blooming (a), Melting (b), Baking (c) and Rotting (d). For each we show the two input frames (col 1-2), and results for: p_mse+adv (col 3), p_g_mse+adv (col 4), p_g_mse+adv+ft (col 5) and ground truth (col 6)

3.3.4 Two stack generation results

For two stack generation, the model generates an image showing the future state of an object given two input depictions at different stages of transformation. As training data, we sample frame triples from videos with neighboring frame degree intervals: 0, 0.1, 0.2, 0.3, 0.4, and 0.5. We train two stack generation models for each of the 4 transformation categories, trained for 12500 iterations for the **p_mse+adv** and **p_g_mse+adv** methods. For the **p_g_mse+adv+ft** method, the models are first pre-trained on the reconstruction dataset for 4000 iterations and then fine-tuned on timelapse data for 6500 iterations (Figure. 3.3 shows example prediction results). We observe that **p_g_mse+adv+ft** generates improved results in terms of both image quality and future state prediction accuracy. We further evaluate the reconstruction accuracy off these models in Table. 3.2 (cols 5-7). Furthermore, in this task we expect that the models can not only predict the future state,

Figure 3.4: Two stack generation with varying time between input images. For each example we show: the two input images (col 1-2), p_mse+adv (col 3), p_g_mse+adv (col 4), p_g_mse+adv+ft (col 5) and ground truth (col 6). The models are able to vary their outputs depending on elapsed time between inputs.

	Pairwise				Two stack			Recurrent		
	BL	ADV	Grad	Ft	ADV	Grad	Ft	ADV	Grad	Ft
Blooming	0.1320	0.2300	0.2880	0.3500	0.3080	0.3240	0.3680	0.3320	0.2860	0.3820
Melting	0.1680	0.2520	0.2760	0.3040	0.3400	0.3120	0.3480	0.3180	0.3220	0.3600
Baking	0.1620	0.2600	0.2840	0.2940	0.3120	0.3200	0.3680	0.2780	0.3540	0.3680
Rotting	0.1340	0.2020	0.2580	0.4060	0.3040	0.2640	0.4320	0.2940	0.2900	0.4160
Average	0.1490	0.2360	0.2765	**0.3385**	0.3160	0.3050	**0.3790**	0.3055	0.3130	**0.3815**

Table 3.3: Human evaluation results: BL stands for p_mse method, ADV (p_mse+adv), Grad(p_g_mse+adv) and Ft(p_g_mse+adv+ft)

but also learn to generate the correct time interval based on the input images. Figure. 3.4 shows input images with different amounts of elapsed time. We can see that the models are able to vary how far in the future to generate based on the input image interval.

3.3.5 Recurrent generation results

For our recurrent generation task, we want to train a model to generate multiple future states of an object given a single input frame. Due to limited data we recursively generate 4 time steps. During training, we sample groups of frames from timelapse videos. Each group contains 5 frames, the first being the input, and the rests having 0, 0.1, 0.2, 0.3 degree intervals from the input. As in the previous tasks, the reconstruction outputs are used for quantitative evaluation.

We train the models separately for the 4 transformation categories. The models for the **p_mse+adv** and **p_g_mse+adv** methods are trained for 8500 iterations on timelapse data from scratch. For the **p_g_mse+adv+ft** method, models are pre-trained on the reconstruction dataset for 5000 iterations then fine-tuned for another 5500 iterations. Figure. 3.5 shows prediction results (outputs of $2^{nd}, 3^{rd}$ and 4^{th} time steps) of the three methods. Table. 3.2 (cols 8-10) shows the reconstruction evaluation.

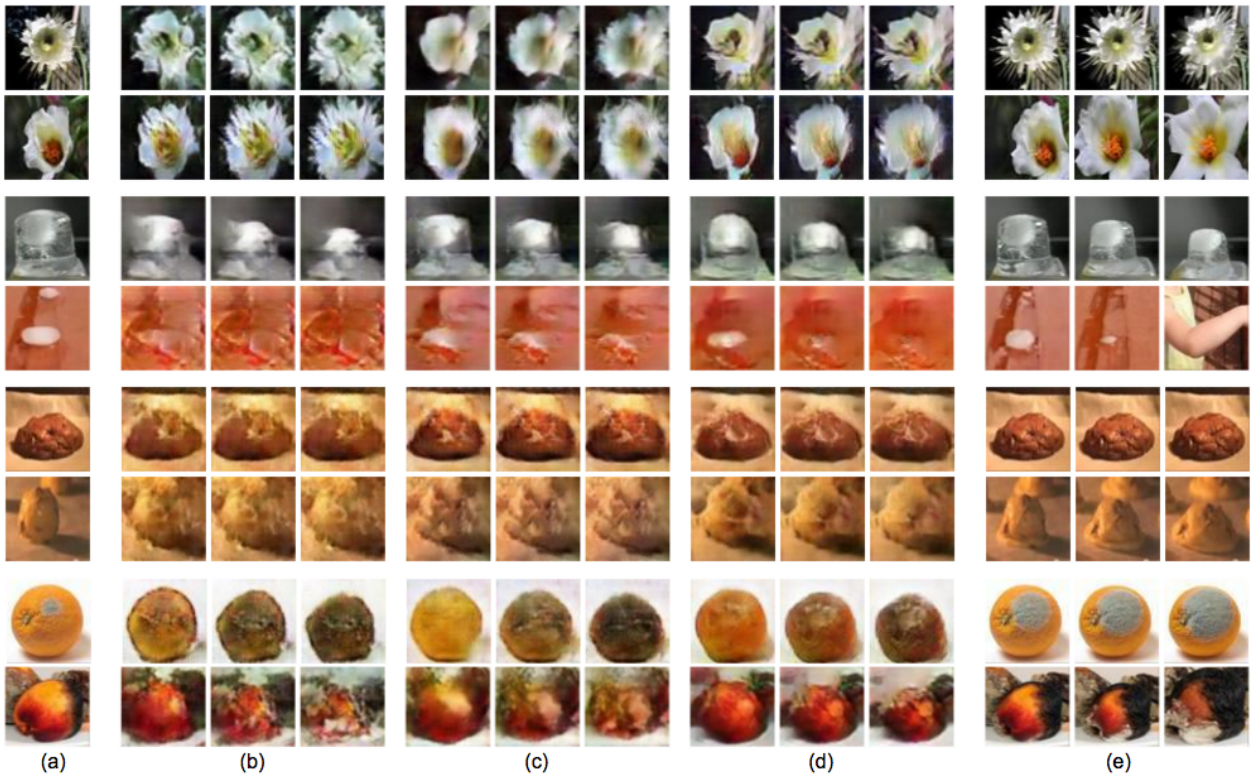


Figure 3.5: Recurrent generation results for: Blooming (rows 1-2), Melting (rows 3-4), Baking (rows 5-6) and Rotting (rows 7-8). Input(a), p_mse+adv(b), p_g_mse+adv(c), p_g_mse+adv+ft(d), Ground truth frames(e).

3.3.6 Additional experiments

Human Evaluations: As previously described, object future state prediction is sometimes not well defined. Many different possible futures may be considered reasonable to a human observer. Therefore, we design human experiments to judge the quality of our generated future states. For each transformation category and generation task, we randomly pick 500 test cases. Human subjects are shown one (or two for two stack generation) input image and future images generated by each method (randomly sorted to avoid biases in human selection). Subjects are asked to choose the image that most reasonably shows the future object state.

Results are shown in Table 3.3, where numbers indicate the fraction of cases where humans selected results from each method as the best future prediction. For pairwise generation (cols 2-5), the **p_g_mse+adv+ft** method achieves the most human preferences, while the baseline performs worst by a large margin. For both two stack generation (cols 6-8) and recurrent generation (cols 9-11), **p_mse+adv** and **p_g_mse+adv** are competitive, but again making use of pre-training plus fine-tuning obtains largest number of human preferences.

Image retrieval: We also add a simple retrieval experiment on Pairwise generation results using pixelwise similarity. We count retrievals within reasonable distance (20% of video length) to the ground truth as correct, achieving average accuracies on top-1/5 of **p_mse+adv**:0.68/0.94; **p_g_mse+adv**: 0.72/0.95; and **p_g_mse+adv+ft**: 0.90/0.98.

Visualizations: Object state transformations often lead to physical changes in the shape of an object. To further understand what our models have learned, we provide some simple visualizations of motion features computed on generated images. Visualizations are computed on results of the **p_g_mse+adv+ft** recurrent model since we want to show the temporal trends of the learned transformations. For each testing case, we compute 3 optical flow maps in the x and y directions between the input image and the second, third, and fourth generated images. We cluster each using kmeans (k=4). Then, for each cluster, we average the optical flow maps in the x and y directions.

Figure. 3.6 shows the flow visualization: (a) is the x axis flow for the blooming transformation. From the visualization we observe the trend of the object growing spatially. (b) shows the y axis flows for the melting transformation, showing the object shrinking in the y direction. (c) shows

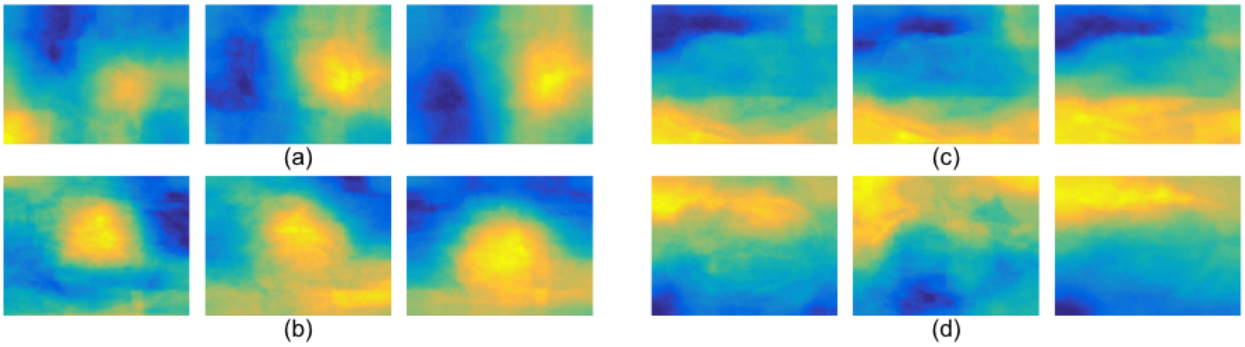


Figure 3.6: Visualization results of learned transformations: x axis flows of blooming (a), y axis flows of melting (b), y axis flows of baking (c) and y axis flows of rotting (d)

baking, consistent with the object inflating up and down. For rotting (d), we observe that the upper part of the object inflates with mold or shrinks due to dehydration.

3.4 Conclusions

In this work, we have collected a new dataset of timelapse videos depicting temporal object transformations. Using this dataset, we have trained effective methods to generate one or multiple future object states. We evaluate each prediction task under a number of different loss functions and show improvements using adversarial networks and pre-training. Finally, we provide human evaluations and visualizations of the learned models. Future work includes applying our methods to additional single-object transformations and to more complex transformations involving multiple objects.

CHAPTER 4

Personalized motion transfer

In this chapter, we develop computational methods for imitating human movements. In particular, we train a model using one several-minute-long video of a target person and can then transfer any desired movement from a new reference video to the target person (maintaining appearance) as Figure 1.2 (Section 1.2) shows. Previous work [Ma et al., 2017, Siarohin et al., 2018, Balakrishnan et al., 2018, Chan et al., 2018] investigates human pose transfer either under generalized settings (models trained across various individuals/scenes) or using single person lab-recorded videos. In this work, we explore personalized motion transfer on videos obtained from the Internet, such as YouTube. With such data pre-recorded in uncontrolled ways, our models are required to generalize well to novel poses, as we cannot ask a person on YouTube to demonstrate the entire range of poses we might want to generate. We show example frames of the Internet videos used as training data in Figure 4.1.

We formulate the motion transfer problem as a pose guided image translation task. Specifically, given an input frame (from the target person) and a desired pose, our model synthesizes the output as the target person in that new pose. Because our goal is to develop a general method that can handle unconstrained target videos from the Internet, which may contain moving background objects or unstable camera motions, we break our task down into two pieces that deal with the foreground (person) and background separately. In particular, our model is composed of two stages: the human synthesis stage which extracts body segments of the target person and then generates a photo-realistic image of the target in a new pose; and a fusion stage which combines the generated person with the extracted background scene. The latter fusion stage helps to remove small artifacts introduced from the human generation stage as well as adding shadows/reflections due to the interaction between the person and the scene. In addition, because we separate foreground and background processing, a side benefit of our method is the ability to place the generated person on a *new background* (something that cannot be achieved with whole frame generation methods). To obtain high-quality synthesis

Figure 4.1: Example frames of personal videos from YouTube (from left to right are: Video_01 to Video_08).

results, we also apply temporal smoothing to generate temporally coherent movements and use hand, foot, and face landmarks to enhance local body details. We describe our method in Section 4.1. To evaluate the effectiveness of the proposed method, we conduct both numerical evaluations and human experiments on generated results in Section 4.2.

4.1 Approach

We formulate the personalized motion transfer task as follows. Given an input frame I_{in} and a reference frame I_{ref} of size $H \times W$, as well as their associated human poses P_{in} and P_{ref}, we would like to learn a mapping $\mathcal{F}$ that generates an output frame I_{out} with the reference pose transferred to the target person in the input frame:

$$I_{out} = \mathcal{F}(I_{in}, P_{in}, P_{ref}) \tag{4.1}$$

where I_{out} retains the same human appearance and background scene as I_{in} while rendering with the reference pose, P_{ref}. During training, we sample random frame pairs from a personal video as our training data (I_{in}, I_{ref}).

We treat the human body part segments from I_{in} as tangram pieces[1] which are placed on a composition image according to the layout of reference pose, P_{ref}. A similar step is used in [Balakrishnan et al., 2018]. This provides us with a good initial image on top of which we apply the human synthesis part of our method. The body part transform process is illustrated in Fig. 4.2 and

[1]A dissection puzzle consisting of fixed number of pieces to form various shape by translation and rotation.

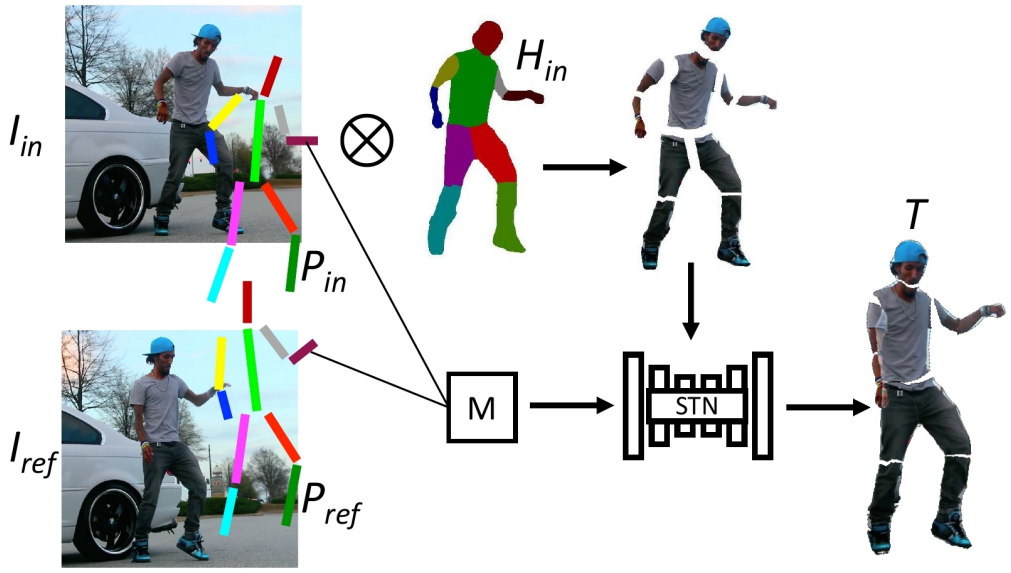


Figure 4.2: Body parts transformation. STN aligns the body parts of I_{in} with target pose P_{ref}, producing body segments represented as a 3D volume T.

described in detail in Section 4.1.1.

Figure 4.3 shows the architecture of our two-stage model. The human synthesis stage (a) takes the transformed body parts as input and produces a foreground body image, filling the "holes" between transformed segments and adjusting details (e.g., angle of face) to produce a photo-realistic image of the target person in the reference pose (Section 4.1.3). In the second stage, a fusion network (b) takes the generated foreground and a fixed background image as inputs and generates the final synthesized output I_{out} (Section 4.1.4). The fusion network combines the foreground and background, fixing discontinuous foreground boundaries and adding important details such as shadows. We design special training techniques and loss functions as discussed in Section 4.1.5.

4.1.1 Human body part transformation

For each frame pair (I_{in}, I_{ref}), we first apply a human body parsing algorithm [Fang et al., 2018] on I_{in} as illustrated in Fig. 4.2, where H_{in} is the human parsing map and $\otimes$ represents element-wise multiplication. The person from the input frame is segmented into 10 body parts (head, torso, left/right upper arms, left/right lower arms, left/right upper legs, left/right lower legs). We also employ the AlphaPose [Fang et al., 2017] pose estimation algorithm to detect 2D poses, (P_{in}, P_{ref}), in both input and reference frames.

Given the pose estimates, we connect line segments between pairs of key points for each of

the ten body parts. Based on the corresponding line segments in P_{in} and P_{ref}, we compute affine transformation matrices $\{M_i \in \mathbb{R}^{2\times 3}\}_{i=1,...,10}$ which define the transformation to align each body part in I_{in} with the pose in reference frame P_{ref}. This transformation helps normalize the body part rotation and scaling between two images (the reference person may appear smaller or larger than the target person). A spatial transformer network [Jaderberg et al., 2015], which applies image warping operations including translation, scale, and rotation, is used to transform input frame body parts according to transformation matrices M_i and generate reference-aligned body segments $T \in \mathbb{R}^{H\times W\times(3\times 10)}$ (Figure 4.2).

4.1.2 Pose map representation

Previous works [Ma et al., 2017, Siarohin et al., 2018] represent pose information as keypoint maps. We find that this can cause 'broken limbs' (missing or disconnected body parts) when synthesizing foreground human body. In our work, we alleviate this issue by encoding the position, orientation and size of each body part as Gaussian smoothed heat map parameterized by a solid circle or rectangle, as illustrated by $\mathbf{P}_{ref}$ in Figure 4.3. The shape parameters (radius, height, width, etc.) are estimated from the associated key points.

Enhancing local details: Besides filling holes between transformed body parts, the human synthesis model is also expected to correct the orientation of head/feet/hands since errors on these important body parts can severely degrade the perceptual quality of results. Thus, in addition to the original 10 body parts, we also add facial, hand and foot landmarks detected by OpenPose [Cao et al., 2017, Simon et al., 2017][2], and encode them in the Gaussian smoothed heat map format. In summary, we represent the reference pose map as a 3D volume $P_{ref} \in \mathbb{R}^{H\times W\times(10+5)}$, with the first 10 channels representing limbs/trunk and the last 5 channels representing facial, left/right hands, left/right feet landmarks.

Temporal smoothing: What we have described so far focuses on image-based frame-to-frame translation. Since our input pose maps are characterized by 2D spatial information, there exists

[2]We use AlphaPose [Fang et al., 2017] as our main detection algorithm because of its robust performance, and use OpenPose [Cao et al., 2017] to gather additional face, hand, and foot key points which are unavailable from AlphaPose.

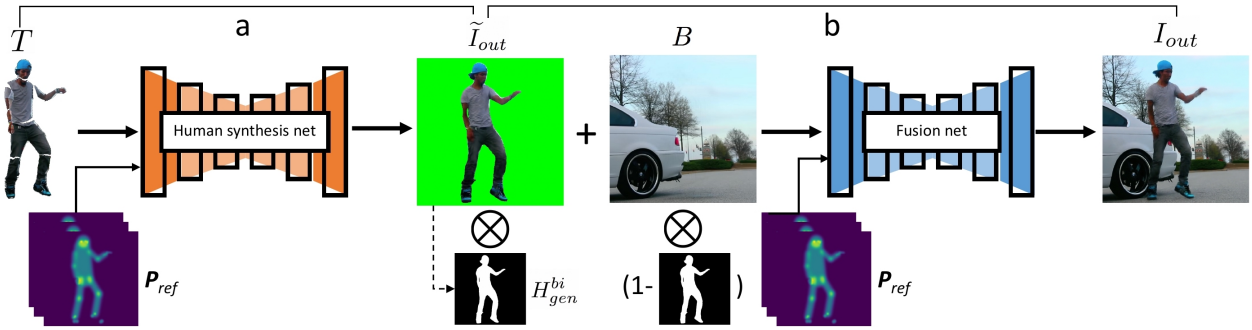


Figure 4.3: Two-stage motion transfer framework. (a) The human synthesis network takes transformed body parts T as input to synthesize human image $\widetilde{I}_{out}$ on a green background, simplifying foreground H^{bi}_{gen} extraction. $\mathbf{P}_{ref}$ represents the aggregated pose maps. (b) The fusion network takes the combined foreground image $\widetilde{I}_{out}$ and fixed background B as input and synthesizes the final output frame I_{out}.

ambiguity when inferring the underlying body configuration in the 3D world. Such uncertainty leads to non-smooth video outputs or temporal flickering. To enforce smooth change across the generated frames, we introduce temporal smoothing through the inputs of our model. We include the previous K reference poses as model inputs. Specifically, we stack $P^{t-k+1}_{ref}, ..., P^{t}_{ref}$ along the pose map channel where P^{t}_{ref} denotes the current reference pose detected from I^{t}_{ref}. Now the stacked pose map $\mathbf{P}_{ref}$ has $(10+5)K$ channels.

4.1.3 Human synthesis network

The human synthesis network generates the foreground person in the reference pose given the transformed body parts T and the pose map $\mathbf{P}_{ref}$ (Section 4.1.2). To train the synthesis network, we convert the body parsing map H_{ref} for reference image I_{ref} [Fang et al., 2018] to a binary mask H^{bi}_{ref}, and extract only the foreground human region $\widetilde{I}_{ref} = I_{ref} \otimes H^{bi}_{ref}$ as training supervision. Of course, the parsed mask will not always be perfect due to algorithmic limitations. Artifacts such as broken limbs will be refined by the fusion network (Section 4.1.4). Note that to avoid distraction by background clutters, the synthesis network is trained to generate human body on a green background with Chroma key compositing as shown in Figure 4.3 (a). This uniform background can simplify foreground extraction so that we can directly obtain the foreground mask H^{bi}_{gen} of the generated frame $\widetilde{I}_{out}$ for later processing. As discussed in Section 4.1.2, the human synthesis net adjusts important pose details, such as head/feet orientations, as exemplified in Fig. 4.4 (a).

4.1.4 Fusion network

The fusion network is used to combine the generated human foreground image $\widetilde{I}_{out}$ with a fixed background image B to generate the final output I_{out}. The background image is obtained by averaging all the frames in the personal video after masking out the foreground with human segmentation results [Fang et al., 2018]. B might contain artifacts such as blurriness due to moving objects or small camera motions that will be refined by the fusion network.

As shown in Fig. 4.3 (b), the fusion network takes the combined foreground/background frame I_{comb} and the stacked pose map $\mathbf{P}_{ref}$ as inputs. The combined frame I_{comb} is given by:

$$I_{comb} = H^{bi}_{gen} \otimes \widetilde{I}_{out} + (1 - H^{bi}_{gen}) \otimes B \tag{4.2}$$

where H^{bi}_{gen} is the foreground mask for the generated human frame $\widetilde{I}_{out}$. We apply these "cut and paste" operations to explicitly define the layering order of the human and background for better blending results.

We use the reference frame I_{ref} as supervision for training the fusion network. Besides blending the human with the background scene, the fusion network helps refine the human (foreground) and fills in broken limbs introduced in the previous stage as demonstrates by the example in Fig. 4.4(b). The example in Fig. 4.4(c) shows the network's ability to add appropriate shadows for interactions between human and scene. In particular, on the left side, we show two frames I_{comb} and I_{out}. In the zoomed-in regions on the right, we observe that the fusion network has removed artifacts from the estimated background B and added shadows consistent with the lighting and person's position.

4.1.5 Loss functions

In order to synthesize frames at high resolution, we utilize a pix2pixHD-style [Wang et al., 2018] framework to train the model in a multi-scale manner. The human synthesis network and fusion network share the same architecture as well as the same loss functions. As usual, G denotes the generator and D denotes the discriminator. We use I to represent input frames I_{in} or I_{comb}, x to represent predicted results $\widetilde{I}_{out}$ or I_{out}, y to represent supervision (ground truth) $\widetilde{I}_{ref}$ or I_{ref}, and p to represent the reference pose maps $\mathbf{P}_{ref}$. So we have $x = G(I, p)$. We employ several losses in our model:

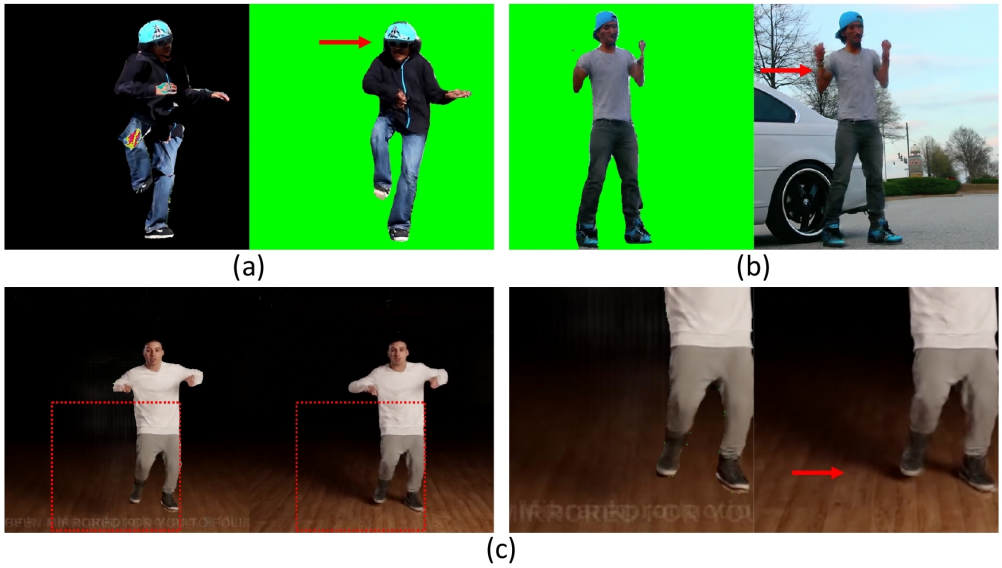


Figure 4.4: (a) The foreground synthesis net adjusts the body segments (left) to obtain correct face orientation (right). (b) The fusion net fills in the background and fixes broken limbs (e.g., arms) produced by the synthesis net. (c) The combined frame I_{comb} and fusion net output I_{out} (left), with their zoomed-in regions (right).

Relativistic average LSGAN loss $\mathcal{L}_{rela}$: The traditional LSGAN [Mao et al., 2017] losses based on our task can be formatted as:

$$\begin{aligned}\min_D \mathcal{L}_{LSGAN}(D) =& \frac{1}{2}\mathbb{E}_{y,p}[(D(y,p)-1)^2]+ \\ & \frac{1}{2}\mathbb{E}_{x,p}[(D(x,p))^2] \\ \min_G \mathcal{L}_{LSGAN}(G) =& \frac{1}{2}\mathbb{E}_{x,p}[(D(x,p)-1)^2]\end{aligned} \tag{4.3}$$

To make training more stable, we make the LSGAN framework relativistic [Jolicoeur-Martineau, 2018]. Here, the main idea is that D estimates the probability that the input is real while G increases the probability that fake data is real. [Jolicoeur-Martineau, 2018] argues that it is necessary for G to also decrease the probability that real data is real.We also apply a gradient penalty [Gulrajani et al., 2017]

term in D. Define $\mu(.)$ as the mean operation, and the relativistic average LSGAN becomes:

$$\begin{aligned}\min_D \mathcal{L}_{rela}(D) =& \frac{1}{2}\mathbb{E}_{x,y,p}[(D(y,p) - \mu(D(x,p)) - 1)^2] + \\ & \frac{1}{2}\mathbb{E}_{x,y,p}[(D(x,p) - \mu(D(y,p)))^2] + \\ & w_{GP}\mathbb{E}_{\hat{x,p}}[(\parallel \bigtriangledown_{\hat{x,p}} D(\hat{x,p}) \parallel_2 -1)^2] \\ \min_G \mathcal{L}_{rela}(G) =& \frac{1}{2}\mathbb{E}_{x,y,p}[(D(x,p) - \mu(D(y,p)) - 1)^2] + \\ & \frac{1}{2}\mathbb{E}_{x,y,p}[(D(y,p) - \mu(D(x,p))^2]\end{aligned} \tag{4.4}$$

where w_{GP} is the weight for the gradient penalty term.

Feature matching loss $\mathcal{L}_{FM}$: We use the feature matching loss from [Wang et al., 2018]. Specifically, we minimize the distance of the features extracted from different layers of D between real and synthesized frames:

$$\mathcal{L}_{FM} = \mathbb{E}_{x,y,p} \sum_{i=1}^{M} \frac{1}{N_i}[\parallel D^i(y,p) - D^i(x,p) \parallel_1] \tag{4.5}$$

where M is the number of layers in D, N_i is the number of elements in each layer, and D^i denotes the i^{th} layer of D.

Perceptual loss $\mathcal{L}_{VGG}$: We utilize the intermediate representations of VGG19 pre-trained on ImageNet classification. We use $\varphi(.)$ to represent the VGG network and the perceptual loss is computed as:

$$\mathcal{L}_{VGG} = \mathbb{E}_{x,y} \sum_{i=1}^{M} \frac{1}{N_i}[\parallel \varphi^i(y) - \varphi^i(x) \parallel_1] \tag{4.6}$$

where φ^i represents i^{th} layer of the VGG network with N_i elements and M is the number of layers.

Semantic layout and pose feature losses $\mathcal{L}_{SP}$: We apply this loss in order to encourage the model to synthesize frames with similar pose and human semantic layouts as ground truth. We use a pre-trained model from [Nie et al., 2018] which jointly performs human semantic parsing and pose estimation. We extract intermediate representations from the pose encoder and parsing encoder, resulting in feature vectors in $\mathbb{R}^{256\times256\times64}$. We use ϕ_p to denote the pose encoding network and ϕ_s

to represent the semantic parsing encoding network. Then, the loss can be formatted as:

$$\begin{aligned}\mathcal{L}_{SP} =& \mathbb{E}_{x,y}[\| \phi_p(y) - \phi_p(x) \|_1] + \\ & w_S \mathbb{E}_{x,y}[\| \phi_s(y) - \phi_s(x) \|_1],\end{aligned} \tag{4.7}$$

where w_S is the weight for the semantic parsing loss.

Finally, we linearly combine all losses as:

$$\mathcal{L} = w_{rela}\mathcal{L}_{rela} + w_{FM}\mathcal{L}_{FM} + w_{VGG}\mathcal{L}_{VGG} + w_{SP}\mathcal{L}_{SP} \tag{4.8}$$

where w's represent the weights for each loss term.

4.2 Experiments

In this section, we introduce training and inference data (Section 4.2.1) as well as the model architectures and training details (Section 4.2.2), show qualitative visualizations (Section 4.2.3), and present numerical and human evaluations (Section 4.2.4 and Section 4.2.5).

4.2.1 Dataset

To study personalized motion transfer on Internet videos, we collect 8 videos from YouTube ranging from 4 to 12 minutes. Figure 4.1 shows some example frames for each video. These personal videos include dancing videos and dancing tutorial videos. For each video, we train a personalized model.

We also collect pose reference videos from different people which are used to drive our personalized motion generation. 16 short videos are downloaded from YouTube including various dance types such as break-dance, shuffle dance and ballet, as well as Taichi martial art which has very different motion style from the target personal videos. All video URLs are provided in the supplementary material.

4.2.2 Model and training details

The human synthesis network and fusion network use the same architecture. For each we apply a multi-scale generator and discriminator. Specifically, we train the model under two scales: 256×256 and 512×512. We use g and $G = (g, \widetilde{g})$ to refer to their generators respectively. $\widetilde{g}$ means the additional convolution layers and residual blocks stacked with the start and end of g, and jointly trained to form G. We also use a two-scale discriminator for lower resolution prediction and

three-scale for higher resolution. Due to GPU memory limitation, we make the number of activation maps of each layer half of the original model [Wang et al., 2018]. During training, we apply Adam Stochastic Optimization [Kingma and Ba, 2014] with learning rate 0.0002 and batch size 4.

We set K=3 for reference pose maps $\mathbf{P}_{ref}$, i.e., stacking the previous two and the current reference poses as network inputs to encourage temporal smoothness. We set the weight of the gradient penalty term w_{GP}=10 and w_S=0.01 in Eq. 4.7. The linear combination weights in Eq. 4.8 are w_{rela}=1, w_{FM}=10, w_{VGG}=10 and w_{SP}=10.

We train one model for each personal video, sampled at 30 FPS. We split each video into training and testing sequences by ratio of 0.85:0.15 (there is no overlap between the training and testing sets) and randomly sample 20,000/2,000 (I_{in}, I_{ref}) frame pairs from the training/testing sub-sequence. This testing set is used for quantitative evaluation because of the availability of ground truth I_{out} (which is from the same person as I_{ref}).

4.2.3 Qualitative results

We visualize the generated results from our model for testing and inference. For each personal video we randomly sample testing pairs that have never been seen by the model during training. In addition to using unseen frames from an input video to drive motion transfer, we can also drive motions of the target person using novel reference videos. Here there is no ground truth to compare with, but we can evaluate results using human experiments. For each reference frame from a new video, we select I_{in} from the personal video with the shortest normalized pose keypoint distance to the reference pose.

We show synthesized results from our method as well as other approaches. We utilize **pix2pixHD** [Wang et al., 2018] as one baseline. This model learns a mapping directly from pose P_{ref} to frame I_{out}. We use the loss functions and model structure from [Wang et al., 2018], but reduce the number of activation maps in each layer by half (as in our model). We also evaluate the **Posewarp** model from [Balakrishnan et al., 2018] as another baseline. Finally, we evaluate a simplified variant of our model **Ours-baseline** that has the same two-stage model architecture but trained without supervision on the human synthesis result $\widetilde{I}_{out}$. For the baseline **Posewarp** method, we generate 256×256 resolution frames as in [Balakrishnan et al., 2018], while results from the others are at 512×512 resolution. We show examples of both testing (I_{ref} and I_{out} of the same person) and inference (I_{ref} and I_{out} of different persons) results in Figure 4.5 and Figure 4.6 respectively. We

Figure 4.5: Generation results on testing set of four methods from the first to fourth row: Posewarp, pix2pixHD, Ours-baseline, Ours. Row 5 shows the zoomed-in regions and the last row shows the reference frames, which are also the ground truth outputs.

Figure 4.6: Inference generation results: we show two sets (column 1-3 and column 4-6) of generation results from our model. For each set, the first two columns are the synthesized results and the last column is a sequence of reference frames from a different video.

compare the proposed method with competing methods on the testing set and show motion transfer from novel reference videos to any of our training videos on the inference set.

Compared with single-stage models which learn direct mapping between poses and video frames, our two-stage model divides this process into easier sub-tasks. This improves generation quality for both foreground and background, and reduces blending artifacts between synthesized human and background scene as Figure 4.7 shows. In this work, the human synthesis and fusion networks are jointly trained while retaining their own respective supervision. One advantage is that the intermediate human generation can be utilized to place the generated individual on new backgrounds. In Figure 4.8, we show several examples where the generated foregrounds have been composited on other scenes from the Internet. We apply simple Gaussian blur on the boundary of the foreground to alleviate border effects and more advanced blending techniques could be used to further improve visual quality.

Figure 4.7: Comparison between results generated by single-stage pix2pixHD (left) and our two-stage model (right).

Figure 4.8: Combine generated foreground person with different scenes.

	MSE	PSNR	SSIM
pix2pixHD [Wang et al., 2018]	702.5714	20.9388	0.7947
Posewarp [Balakrishnan et al., 2018]	744.3939	20.6992	0.7663
Ours-baseline	664.7313	21.2318	0.8064
Ours	642.9080	21.3286	0.8115

Table 4.1: Evaluation of whole frame synthesis results on testing set.

	MSE	PSNR	SSIM
pix2pixHD [Wang et al., 2018]	191.5246	25.7823	0.9314
Posewarp [Balakrishnan et al., 2018]	191.1796	25.8334	0.9264
Ours-baseline	176.9530	26.0477	0.9338
Ours	171.3259	26.1752	0.9352

Table 4.2: Evaluation of foreground synthesis results on testing set.

4.2.4 Numerical evaluation

We quantitatively evaluate testing results of the proposed method and other approaches with metrics Mean Square Error (MSE), PSNR, and SSIM averaged over all personal videos. The results for the whole frames and foreground (human) regions are reported in Table 4.1 and 4.2, respectively. Our method achieves the best synthesis quality in MSE, PSNR, and SSIM for both full-frame and foreground-only evaluations. The Posewarp method achieves relatively high errors in whole frame evaluation because the moving background in some videos violates the static background assumption of the method (background synthesized through hole-filling on the input frame).

To measure the temporal coherence of generated frames, we calculate the difference between all consecutive generated frame pairs ($I^t_{out}-I^{t-1}_{out}$) from the testing results, and measure the MSE against the ground truth difference frame. This simple metric can be regarded as a rough surrogate to optical flow difference. As can be seen from the results averaged over all the input videos in Table 4.3, both our full model and ours-baseline achieve competitive temporal coherence on the synthesized results.

	Whole frame	Foreground
pix2pixHD [Wang et al., 2018]	225.7470	103.5557
Posewarp [Balakrishnan et al., 2018]	220.8329	103.4395
Ours-baseline	219.5680	98.9921
Ours	217.8404	99.5027

Table 4.3: The MSE of difference frame on testing set.

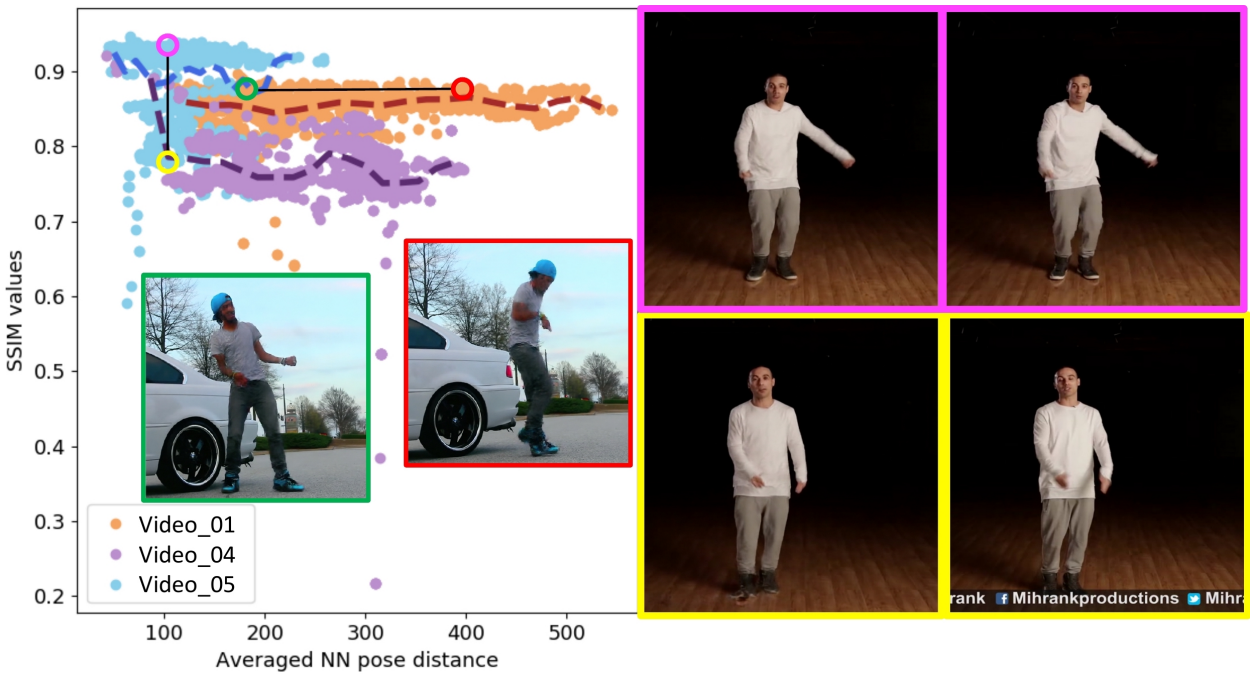


Figure 4.9: Relationship between pose distance to training samples and generation quality. The highlighted frame pair with magenta and yellow color (left is generated and right is ground truth) have the same pose distance with training sample but different SSIM scores due to the subtitle occasionally observed in the background of the yellow frame. The frame pair of green and red color have similar generation quality although in training the model has only seen similar poses as the green one but not the red one.

Generalization to unseen reference poses, e.g. to transfer ballet motion to a target person whom has only been observed to perform hip-hop dancing, is critical for our personal model as it is trained with an Internet video with only a few minutes covering a limited portion of human pose space. Figure 4.9 shows the relationship between the "novelty" of reference poses and the SSIM generation quality. Specifically, we compute the normalized keypoints distance between the reference pose and all training poses, and take the average of the top 10 nearest distances as the value in x axis. The testing results from 3 personal videos are plotted as colored dots in Figure 4.9. We observe that even for poses very different from training (large x axis), our model's performance remains stable. Most outliers in Figure 4.9 come from unexpected ground truth such as frames in the video prologue with very dark color.

	pix2pixHD	Posewarp	Ours-baseline	Ours
Q1	14.70%	17.59%	25.98%	41.73%
Q2	13.39%	21.00%	25.20%	40.42%
Q3	17.85%	22.83%	23.36%	35.96%
Q4	19.42%	22.05%	28.87%	29.66%
Q5	14.17%	19.69%	25.98%	40.16%

Table 4.4: Human rating percentages over different methods on complete body (Q1); clearest face (Q2); isolated foreground and background (Q3); temporal stability (Q4); and overall quality (Q5).

4.2.5 Human evaluation

Finally, we also measure the human perceptual quality of generated results, especially for motion transferred from novel videos to a target person, where there exists no ground truth for comparison. Therefore, we conduct a human subjective test to measure motion transfer quality.

In these experiments, we compare videos generated with our proposed method, our simplified baseline model, Posewarp, and pix2pixHD by conducting a forced-choice task on Amazon Mechanical Turk (AMT). We show Turkers videos generated by each of the four methods with random order. Turkers are provided with five questions and asked to select one video as the answer to each question. The questions are related to: 1) the most complete human body (least missing body parts or broken limbs); 2) the clearest face; 3) the most isolated human and scene (the foreground and background are not mixed together); 4) the most temporally stable video (least jitters); 5) the most overall visual appealingness. For each question, we show instructional examples to help the Turker better understand the task. Three different Turkers are asked to label each group of videos for each question, and their selections are aggregated across tasks as the final result. We show the selection rate of the four methods averaged over the 8 personalized models in Table 4.4. We can see that consistent with our previous quantitative ground truth based measurements, the human evaluation also favors our proposed method on all the five questions. In particular, our proposed method achieves much better performance on body completeness, face clarity, foreground/background separation, and the overall visual appearance. For temporal stability, both our full model and baseline model outperform the existing methods.

4.3 Conclusions

In this chapter, we introduced a model for personalized motion transfer on Internet videos. Our model consists of two parts: a human synthesis network and a fusion network. The former synthesizes

a human foreground based on pose-transformed body parts. The latter fuses the person with a background scene and further refines the synthesis details. The evaluation shows that our method can generate personal videos of new motion with visually appealing quality and fewer artifacts than existing methods. Future directions include handling online videos with drastic camera motion (with background significantly changing from frame to frame) and motion transfer using incomplete or partially observed body parts.

CHAPTER 5

Visual to sound

The visual and auditory senses are arguably the most important channels through which humans perceive their surrounding environments, and they are often entertwined. From life-long observations of the natural world, people are able to learn the association between vision and sound. For instance, when seeing a flash of lightning in the sky, one might cover their ears subconsciously, knowing that the crack of thunder is coming next. Alternatively, hearing leaves rustling in the wind might conjure up a picture of a peaceful forest scene.

In this chapter, we explore whether computational models can learn the relationship between visuals and sound. Models of this relationship could be fundamental for many applications such as combining videos with automatically generated ambient sound to enhance the experience of immersion in virtual reality; adding sound effects to videos automatically to reduce tedious manual sound editing work; Or enabling equal accessibility by associating sound with visual information for people with visual impairments (allowing them to "see" the world through sound). While all of these tasks require powerful high-level inference and reasoning ability, in this work we take a first step toward this goal, narrowing down the task to generating audio for video based on the viewable content.

Specifically, we train models to directly predict raw audio signals (waveform samples) from input videos. The models are expected to learn associations between generated sound and visual inputs for various scenes and object interactions. Existing works [Owens et al., 2016, Chen et al., 2017] handle visual to sound task under experimental settings. In our work, we deal with generating natural sound from videos collected in the wild.

To enable learning, we introduce a dataset that is derived from AudioSet [Gemmeke et al., 2017]. AudioSet is a dataset collected for audio event recognition but not ideal for our task because many of videos and audios are loosely related; the target sound might be covered by other sounds (like music); and the dataset contains some mis-classified videos. All of these sources of noise tend to

deter the models from learning the correct mapping from video to audio. To alleviate these issues, we clean a subset of the data, including sounds of humans/animals and other natural sounds, by verifying the presence of the target objects for both videos and audios respectively (at 2 second intervals) to make them suitable for the generation task (Section 5.1).

Our model learns a mapping from video frames to audio using a video encoder plus sound generator structure. For sound generation, we use a hierarchical recurrent neural network proposed by [Mehri et al., 2016]. We present 3 variants to encode the visual information, which can be combined with the sound generation network to form a complete framework (Section 5.2). To evaluate the proposed models and the generated results, we conduct both numerical evaluations and human experiments in Section 5.3.

5.1 Visually Engaged and Grounded AudioSet (VEGAS)

The goal of this work is to generate realistic sound based on video content and simple object activities. Currently, we do not explicitly handle high-level visual reasoning during sound prediction. For the training videos, we expect visual and sound are directly related (predicting dog sound when seeing a dog) most of the time.

Most existing video datasets [Abu-El-Haija et al., 2016, Karpathy et al., 2014] include both video and audio channels. However, they are typically intended for visual understanding tasks, thus organized based on visual entities/events. A better choice for us is AudioSet [Gemmeke et al., 2017], a large-scale object-centric video dataset organized based on audio events. Its ontology includes events such as fowl, baby crying, engine sounds. Audioset consists of 10-second video clips (with audios) from Youtube. The presence of sounds has been manually verified. But as a dataset designed for audio event detection, AudioSet still cannot perfectly fit our needs because of the following three reasons. First, visual and sound are not necessarily directly related. For instance, sometimes the source of a sound may be out of frame, e.g. we may hear a dog barking while the video shows a group of people laughing. Second, the target sound (say mew from a cat) might have been largely covered by other noise like background music. Third, mis-classification exists.

We ran several baseline models using the original data and found that the generated sounds are not clean and often accompanied with other noise like chaotic human chatting. To make the data useful for our generation task, we select a subset of videos from AudioSet and further clean them.

5.1.1 Data collection

We select 10 categories from AudioSet (each including more than 1500 videos) for further cleaning. The selected data includes human/animal sound as well as ambient sounds (specifically they are: Baby crying, Human snoring, Dog, Water flowing, Fireworks, Rail transport, Printers, Drum, Helicopter and Chainsaw). For the categories containing more data than needed, we randomly select 3000 videos for each.

We use AMT, asking turkers to verify the presences of an object/event of interest for the video clip in both the visual and audio modalities. If both modalities are verified we consider it a clean video. For most of the videos, noise does not dominate for the entire videos. Therefore, to retain as much data as possible, we segment each video into 2-second short clips for separate labeling. For each short clip, we divide the video and audio for independent annotation.

To clean the audio modality, we ask turkers to annotate the presence of a sound from a target object (e.g. flowing water for water flowing category). Turkers are provided with three choices: 'Yes — the target sound is dominant over other sounds', 'Sort of — competitive, 'No — other sounds are dominant'. To clean the visual modality, we similarly ask turkers to annotate the presence of a target object with three choices: 'Yes — the target objects appear all the time', 'Sort of — appearing partially', 'No — the target objects do not appear or barely appear'. For each segment, we collect annotations from 3 turkers and pick the majority as the final annotation. We reject turkers with low accuracy to ensure annotation quality.

We remove the clips where either video and audio has been labeled as 'No' and keep the 'Yes' and Sort of' labeled clips to introduce more variation in the collected data. Finally, we combine the verified adjacent short clips to form longer videos, resulting in videos ranging from 2-10 seconds.

5.1.2 Data statistics

In total, we annotated 132,209 clips in both the visual and audio modalities, each labeled by 3 turkers, and removed 34,392 clips from the original data. After merging adjacent short clips, we have 28,109 videos in total with an average length of 7 seconds and a total length of 55 hours. The left table in Fig. 5.1 shows the number of videos and the average length with the standard deviation for each category respectively. The pie chart demonstrates the distribution of lengths, showing that the majority of videos are longer than 8 seconds. Fig. 5.2 shows some example frames with their corresponding waveforms. We can see how sound correlates with the motion of target objects as

	Number	Length (sec)
Dog	2785	6.3±3.2
Chainsaw	1824	7.3±3.2
Water flowing	2924	7.0±3.2
Rail transport	3259	7.8±3.0
Fireworks	3115	7.0±3.1
Printer	3673	7.2±3.2
Helicopter	3718	7.8±3.0
Snoring	2144	5.9±3.3
Drum	2606	7.4±3.2
Baby crying	2061	6.0±3.1

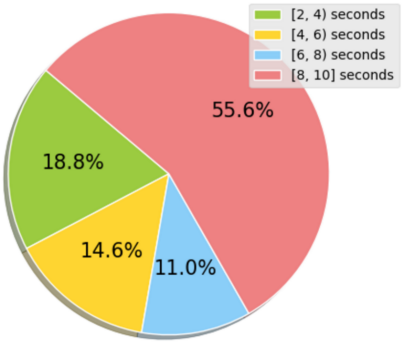


Figure 5.1: Dataset statistics: the table shows the number of videos with averaged length for each category, while the pie chart presents the distribution of video lengths.

well as scene events, such as water flowing (bottom right) where the ambient sounds are temporally uniform. Due to the verified properties of the current dataset, we call it the Visually Engaged and Grounded AudioSet (VEGAS).

5.2 Approaches

In this work, we formulate the task as a conditional generation problem, for which we train a conditional generative model to synthesize raw waveform samples from an input video. Specifically, we estimate the following conditional probability:

$$p(y_1, y_2, ..., y_n | x_1, x_2, ..., x_m) \tag{5.1}$$

where $x_1, ..., x_m$ represent input video frame representations and $y_1, ..., y_n$ are output waveform values which is a sequence of integers from 0 to 255 (the raw waveform samples are real values ranging from -1 to 1, we rescale and linearly quantize them into 256 bins in our model see Section 5.2.1). Note that typically $m << n$ because the sampling rate of audio is much higher than that of video, thus the audio waveform sequence is much longer than video frame sequence for a synchronized video.

We adopt an encoder-decoder architecture in model design and experiment with three variants of this type. In general, our models consist of two parts: video encoder and sound generator. In the following sections, we first discuss the sound generator in Sec. 5.2.1, then we talk about three different variations of encoding visual information and the concrete systems in Sec. 5.2.2, Sec. 5.2.3

Figure 5.2: Video frames of 4 categories from the VEGAS dataset with their corresponding waveforms. The color of the image borders is consistent with the mark on the waveform, indicating the position of the current frame in the whole video.

and Sec. 5.2.4.

5.2.1 Sound generator

Our goal is to directly synthesize waveform samples with a generative model. As mentioned before, in order to obtain audios of reasonable quality (i.e., sounds natural), we adopt a high sampling rate at 16kHz. This requirement results in extremely long sequences, which poses challenges to a sound generator. For this purpose, we choose the recently proposed SampleRNN [Mehri et al., 2016] as our sound generator. SampleRNN is a hierarchically structured recurrent neural network. Its coarse-to-fine structure enables the model to generate extremely long sequences and the recurrent structure of each layer captures the dependency between distant samples. SampleRNN has been applied to speech synthesis and music generation tasks previously. Here we apply it to generate natural sound for videos in the wild, which typically contain much larger variations, less structural patterns, and more noise than speech or music data.

Specifically, Fig. 5.3(a) (upper-left corner brown box) shows the simplified overview of the SampleRNN model. Note, this simplified illustration shows 2 tiers, but more tiers are possible (we use 3). This model consists of multiple tiers, the fine tier (bottom layer) is a multilayer perceptron (MLP) which takes the output from the next coarser tier (upper layer) and the previous k samples

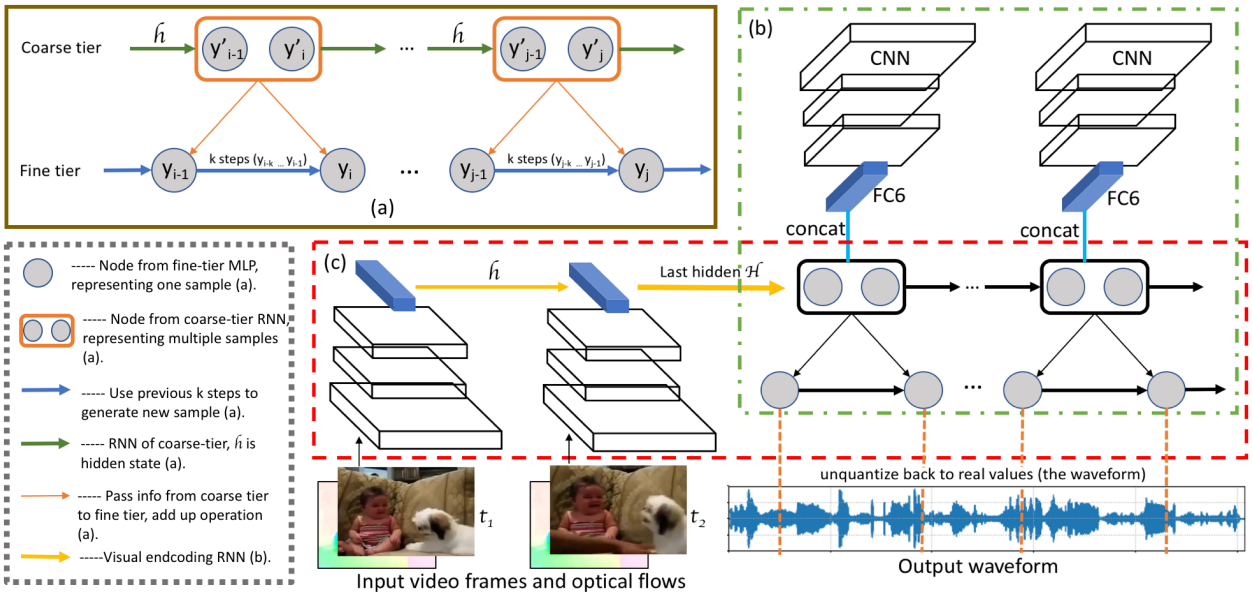


Figure 5.3: (a) (brown box) shows the simplified architecture of the sound generator, where the fine tier MLP takes as input k previously generated samples and output from the coarse tier to guide generation of new samples. (b) (green dotted box) presents the frame-to-frame structure, where we concatenate the visual representation (the blue FC6 cuboid) with the nodes from the coarsest tier. And (c) (red dotted box) shows the model architecture for sequence-to-sequence and flow-based methods, we recurrently embed visual representations and use the last encoding hidden state (the bold yellow arrow) to initialize the hidden state of the coarsest tier RNN of the sound generator. The MLP tier of the sound generator does 256-way classification to output integers within $[0, 255]$, which are linearly mapped to raw waveforms $[-1, 1]$. The legends in the bottom-left gray dotted box summarize the meaning of the visualization units and the letters in the end ((a)/(b)/(c)) point to the part where the unit can be found.

to generate a new sample. During training, the waveform samples (real numbers from -1 to 1) have been linearly quantized to integers ranging from 0 to 255, and the MLP of the finest tier can be considered a 256-way classification to predict one sample at each timestep (then mapped back to real values for the final waveform). The coarser (upper) tiers are recurrent neural networks which can be a GRU [Cho et al., 2014], LSTM [Hochreiter and Schmidhuber, 1997], or any other RNN variants, and the nodes contain multiple waveform samples (2 in this illustration), meaning that this layer predicts multiple samples jointly at each time step based on previous time steps and predictions from coarser tiers. The green arrow represents the hidden state. Note that we tried using the model from WaveNet [van den Oord et al., 2016] on the natural sound generation task, but it sometimes failed to generate meaningful sounds for categories like dog, and was outperformed by SampleRNN consistently for all object categories. Therefore, we did not pursue it further. Due to space limit, we omit the technical details of SampleRNN. For more information regarding SampleRNN, please refer to [Mehri et al., 2016].

5.2.2 Frame-to-frame method

For the video encoder component, we first propose a straight-forward frame-to-frame encoding method. We represent the video frames as $x_i = V(f_i)$, where f_i is the i^{th} frame and x_i is the corresponding representation. Here, $V(.)$ is the operation to extract the $fc6$ feature of VGG19 network [Simonyan and Zisserman, 2014b] which has been pre-trained on ImageNet [Deng et al., 2009] and x_i is a 4096-dimensional vector.

In this model, we encode the visual information by uniformly concatenating the frame representation with the nodes (samples) of the coarsest tier RNN of the sound generator as shown in Figure 5.3(b) (content in dotted green box). Due to the difference of sampling rates between the two modalities, to maintain the alignment between them, for each x_i, we duplicate it s times, so that visual and sound sequences have the same length. Here $s = ceiling[sr_{audio}/sr_{video}]$, where sr_{audio} is the sampling rate of audio, sr_{video} is that of video. Note that we only feed the visual features into the coarsest tier of SampleRNN because of the importance of this layer as it guides the generation of all finer tiers as well as for computational efficiency.

5.2.3 Sequence-to-sequence method

Our second model design has a sequence to sequence type of architecture [Sutskever et al., 2014]. In this sequence-to-sequence model, the video encoder and sound generator are clearly separated,

and connected via a bottleneck representation, which feeds encoded visual information to the sound generator. As Fig. 5.3(c) (content in the middle red dotted box) shows, we build a recurrent neural network to encode video features. Here the same deep feature ($fc6$ layer of VGG19) is used to represent video frames as in Sec. 5.2.2. After visual encoding (i.e., deep feature extraction and recurrent processing), we use the last hidden state from the video encoder to initialize the hidden state of the coarsest tier RNN of the sound generator, then sound generation starts. Therefore the sound generation task becomes:

$$p(y_1, ..., y_n | x_1, ..., x_m) = \prod_{i=1}^{n} p(y_i | H, y_1, ..., y_{i-1}) \tag{5.2}$$

where H represents the last hidden state of the video encoding RNN or equivalently the initial hidden state of the coarsest tier RNN of the sound generator.

Unlike the frame based model mention above, where we explicitly enforce the alignment between video frames and waveform samples. In this sequence-to-sequence model, we expect the model to learn such alignment between the two modalities through encoding and decoding.

5.2.4 Flow-based method

Our third model further improves the visual representation to better capture the content and motion in input videos. As motivation for this variant, we argue that motion signals in the visual domain, even though sometimes subtle, are critical to synthesize realistic and well synchronized sound. For instance, the barking sound should be generated at the moment when the dog opens its month and maybe the body starts to lean forward. This requires our model to be sensitive to activities and motion of target objects. However, the previously used VGG features are pre-trained on object classification tasks, which typically result in features with rotation and translation invariance. Although the VGG features are computed along consecutive video frames, which implicitly include some motion signals, it may still fail to capture them.

Therefore, to explicitly capture the motion signal, we add an optical flow-based deep feature to the visual encoder and call this method the flow-based method. The overall architecture of the current method is identical to the sequence-to-sequence model (as Fig. 5.3(c) shows), which encodes video features x_i recurrently through RNN and decodes with SampleRNN. The only difference is that here $x_i = cat[V(f_i), F(o_i)]$ ($cat[.]$ indicates concatenation operation); o_i is the optical flow of

i^{th} frame; and F(.) is the function to extract the optical flow-based deep feature.

We pre-compute optical flow between video frames using [Sun et al., 2010] and feed the flows to the temporal ConvNets from [Simonyan and Zisserman, 2014a], which has been pre-trained on optical flows of UCF-101 video activity dataset [Soomro et al., 2012], to get the deep feature. We extract the $fc6$ layer of temporal ConvNets, a 4096-dimensional vector.

5.3 Experiments

In this section, we first introduce the model structure and training details (Sec. 5.3.1). Then, we visualize the generated audio to qualitatively evaluate the results (Sec. 5.3.2). Quantitatively, we report the loss values for all methods and evaluate generated results on a video retrieval task (Sec. 5.3.3). Additionally, we also run 3 human evaluation experiments to subjectively evaluate the results from the different proposed models (Sec. 5.3.4).

5.3.1 Model and training details

We train the 3 proposed models on each of the 10 categories of our dataset independently. All training videos have been padded to the same length (10 secs) by duplicating and concatenating up to the target length. We sample the videos at 15.6 FPS (156 frames for 10 seconds) and sample the audios at approximately 16kHz, specifically 159744 times per 10 seconds. For the frame based method, step size s is set to 1024.

Sound generator: We apply a 3-tier SampleRNN with one-layer RNN for the coarsest and second coarsest tiers, and a MLP for the finest tier. For the finest tier, new sample generations are based on the previous k generated samples ($k = 4$). We use GRU as the recurrent structure. The number of samples included by each node from coarse to fine tiers are: 8, 2 ,1 with hidden state size of 1024 for the coarsest and second coarsest tiers.

Frame-to-frame model ($Frame$): To concatenate the visual feature (4096-D) with the nodes from the coarsest tier GRU, we first expand the node (8 samples) to 4096 by applying a fully connected operation. After combining with the visual feature, we obtain a 8192-D vector to feed into the coarsest tier of the sound generator.

Sequence-to-sequence (Seq) & flow based model ($Flow$): These two models have the same architecture. For the visual encoding recurrent neural network, we also use an one-layer-GRU

structure with the hidden state size equal to 1024. The only difference is that for the flow based model, the visual feature is the concatenation of the deep image feature (4096-D) and deep flow feature (4096-D) resulting in a 8192-D vector.

We randomly select 128 videos from each category for testing, leaving the remaining videos for training. No data augmentation has been applied. During training, we apply Adam Stochastic Optimization [Kingma and Ba, 2014] with learning rate 0.001 and minibatch of size 128 for all models. For our experiments we train models for each category independently. As an additional experiment, aiming to handle multiple audio-visual objects within the same video, we also train a multi-category model where we combine data from all categories. We show some results of the multi-category model on videos from the Internet containing multiple interacting objects in the supplementary video.

5.3.2 Qualitative Visualization

We visualize the generated waveform results from the three proposed models as well as the original audio and corresponding video frames in Fig. 5.4. Results from four categories are shown from left to right: Dog, Fireworks, Drum, and Rail transport. The former three are synchronization-sensitive categories, though that doesn't mean the waveform needs to be exactly aligned with the ground truth for good human perception. For instance in the fireworks example (second left), we show the waveforms from $Frame$, Seq, $Flow$ methods and the real audio from top to bottom. Compared to the real audio, the $Flow$ waveform (third) shows several extra light explosions (high peaks). When we listen to it, these extra peaks sound like far away explosions, which reasonably fits the scene. The Rail transport category is not that sensitive to the specific speed of the objects but some of the videos like the depicted example have the obvious property that the amplitude of the sound is affected by the distance of the target object (when the train approaches, the sound gets louder). All three of our models can implicitly learn this effect. We show more qualitative results in the supplementary video.

5.3.3 Numerical evaluation

In this section, we provide quantitative evaluations of the models.

Loss values: First we show the average cross-entropy loss (the finest layer of sound generator does 256-way classification for each sample prediction) for training and testing of $Frame$, Seq and $Flow$

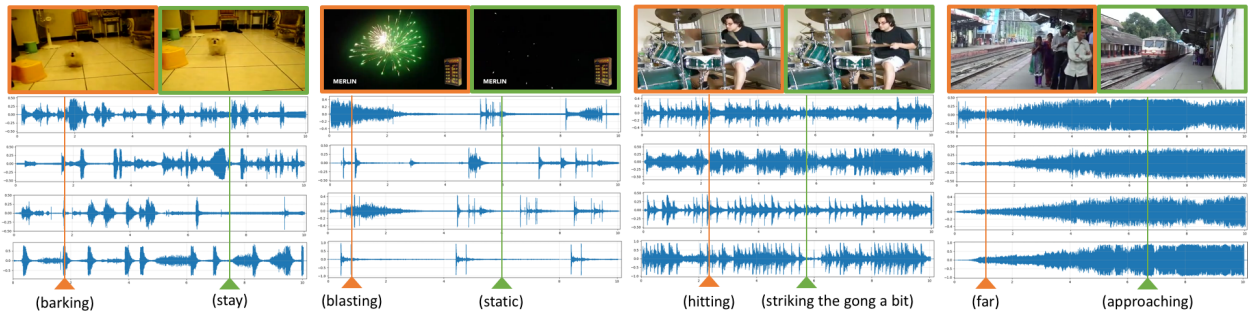


Figure 5.4: Waveforms of generated audio aligned with corresponding video key frames. From left to right showing: Dog, Fireworks, Drum, and Rail transport categories. For each case the 4 waveforms (from top to bottom) are from *Frame*, *Seq*, *Flow* methods, and the original audio. The border color of the frames indicates which flagged position is shown and descriptions indicate what is happening in the video at that moment.

	Frame	Seq	Flow
Training	2.6143	**2.5991**	2.6037
Testing	2.7061	2.6866	**2.6839**

Table 5.1: Average cross-entropy loss for training and testing of 3 methods. Frame represents frame-to-frame method; Seq means sequence-to-sequence method and Flow is flow based method. We mark the best results in bold.

models in Table 5.1. We can see that *Flow* and *Seq* methods achieve lower training and testing loss than *Frame* method, and they are competitive. Specifically *Seq* method has the lowest training loss after converging, while *Flow* works best on testing loss.

Retrieval experiments: Since direct quantitative evaluation of waveforms is quite challenging, we design a retrieval experiment that serves as a good proxy. Since our task is to generate audio given visual representations, well trained models should have the capability of mapping visual information to their corresponding correct (or reasonable) audio signal. To evaluate our models in this direction, we design a retrieval experiment where visual features are used as queries and audio with the maximum sampling likelihood is retrieved (we apply the trained models to map the queries to the audio with lowest loss). Here the audios from all testing videos are combined into a database of 1280 audios, and audio-retrieval performance is measured for each testing video.

If our models have learned a reasonable mapping, the retrieved audio should be (1) from the same category as the query video (category-level retrieval), and more ideally (2) the exact audio corresponding to the query video (instance-level retrieval). Note that this can be very challenging

		Frame	Seq	Flow
Category	Top1	40.55%	44.14%	**45.47%**
	Top5	53.59%	58.28%	**60.31%**
Instance	Top1	4.77%	5.70%	**5.94%**
	Top5	7.81%	9.14%	**10.08%**

Table 5.2: Top 1 and top 5 audio retrieval accuracy. 'Category' measures whether the audio retrieved is of the correct category, while 'Instance' indicates whether the audio retrieved is an exact match to the input video.

since videos may contain very similar contents. In Table. 5.2 we show the average top1 and top5 retrieval accuracy for category and instance retrieval. We observe that all methods are significantly better than chance (where chance for category retrieval is 10% and for instance retrieval is 0.78%). The flow based method achieves the best accuracy under both metrics.

5.3.4 Human Evaluation Experiments

As in image or video generation tasks, the quality of generated results can be very subjective. For instance, sometimes even though the generated sound or waveforms might not be very similar to the ground truth (the real sound), the generation may still sound like a reasonable match to the video. This is especially true for ambient sound categories (e.g. water flow, printer) where the overall pattern may be more important than the specific frequencies, etc. Thus, comparing generated sounds with ground truth by applying distance metrics might not be the ideal way to evaluate quality. In this section, we directly compare the sound generation results from each proposed method in three human evaluation experiments on AMT.

Methods comparison task: This task aims to directly compare the sounds generated by the three proposed methods in a forced-choice evaluation. For each test video, we show turkers the video with audio generated by: the *Frame*, *Seq* and *Flow* methods. Turkers are posed with four questions and asked to select the best video-audio pair for each question. Questions are related to: 1) the correctness of the generated sound (which one sounds most likely to come from the visual contents); 2) which contains the least irritating noise; 3) which is best temporally synchronized with the video; 4) which they prefer overall. Each question for each test video has been labeled by 3 different turkers and we aggregate their votes to get the final results.

Table 5.3 shows the average preference rate for all categories (the higher the better) on each

	Frame	Seq	Flow
Correctness	29.74%	34.92%	**35.34%**
Least noise	28.65%	35.31%	**36.04%**
Synchronization	28.57%	34.37%	**37.06%**
Overall	28.52%	34.74%	**36.74%**

Table 5.3: Human evaluation results in a forced-choice selection task. Here we show the average selection rate percentage over all categories for each of 4 questions (highest selection rate for each question marked in bold).

question. We can see that both *Seq* and *Flow* outperform *Frame* based method with *Flow* performing best overall. *Flow* outperforms *Seq* the most on question 3 which demonstrates that adding the deep flow feature helps with improving the temporal synchronization of visual and sound during generation. Figure 5.5 shows the results for each category. We observe that the advantage of the *Flow* method is mainly gained on categories that are sensitive to synchronization, such as Fireworks and Drum (see question 3 and question 2 histograms).

Visual relevance task: Synchronization between video and audio can be one of the fundamental factors to measure the realness of sound generated from videos, but synchronization tolerance can vary between categories. For example, we easily detect discordance when a barking sound fails to align to the correct barking motion of a dog. While for other categories like water flowing, we might be more tolerant, and may not notice if we swapped the audio from one river to a another. Inspired by this observation, we design a task in which each test video is combined with two audios, and ask the turkers to pick the video-audio pair that best corresponds. One of the audios is generated from the video, while the other is randomly chosen from another video of the same category. This task measures whether the audio we generate is discriminative for the input video.

Table 5.4 shows the percentage of matched audios being correctly selected. Results are reported for the sounds generated by our three methods (the first three columns) as well as the real sounds (the last column). Each test sample is rated by three turkers. The results are consistent with our intuition – real sounds are generally discriminative for the corresponding videos, though some categories like helicopter or water are less discriminative. Our three methods achieve reasonable accuracy in some categories like Dog and Fireworks (outperforming 50% chance by a large margin). For more ambient sound categories, the discrimination task is challenging for both generated and

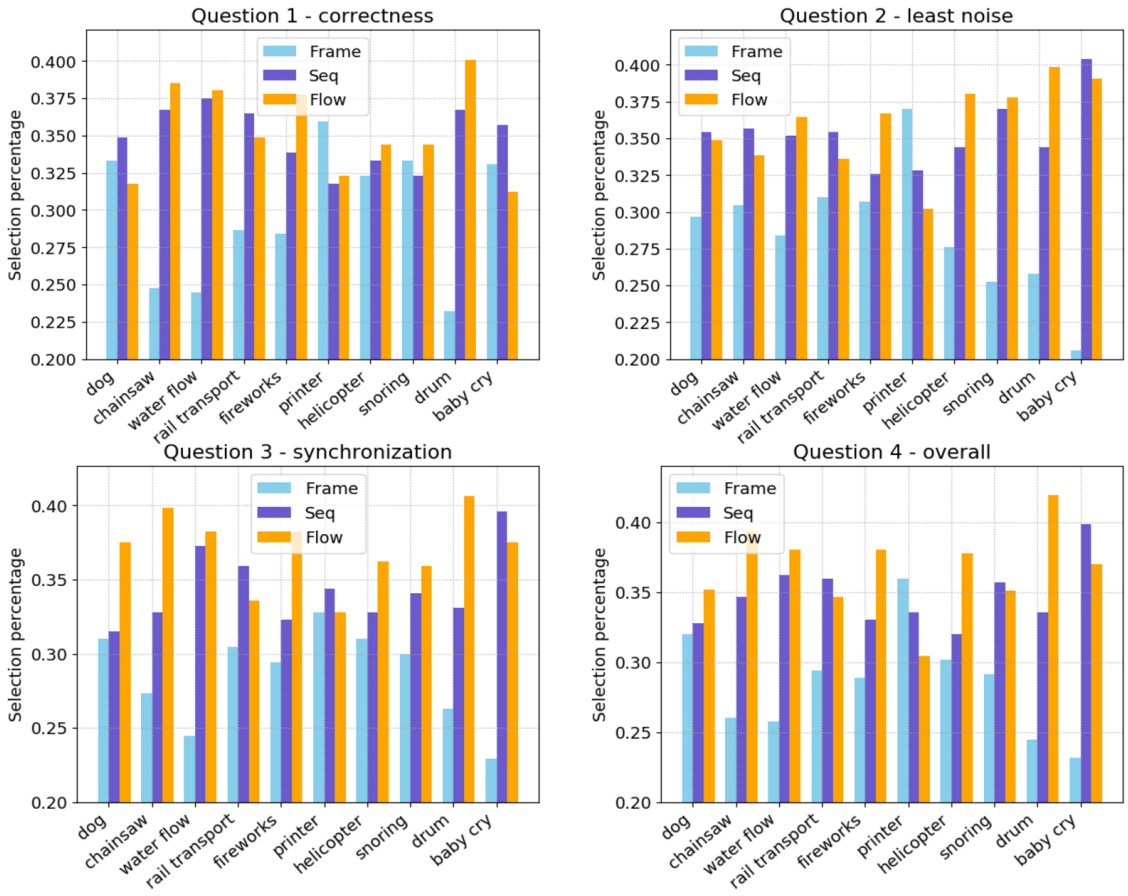


Figure 5.5: Human evaluation of forced-choice experiments for four questions broken down by category.

	Frame	Seq	Flow	Real
Dog	57.29%	58.85%	**63.02%**	75.00%
Chainsaw	56.25%	57.03%	**58.07%**	70.31%
Water flowing	49.22%	52.34%	**52.86%**	59.90%
Rail transport	53.39%	**56.51%**	55.47%	66.14%
Fireworks	61.98%	67.97%	**68.75%**	79.17%
Printer	46.09%	**50.52%**	47.14%	60.16%
Helicopter	51.82%	**54.95%**	54.17%	58.33%
Snoring	51.82%	53.65%	**54.95%**	63.02%
Drum	55.21%	59.38%	**62.24%**	73.44%
Baby crying	52.60%	**57.55%**	56.77%	70.57%
Average	53.56%	56.88%	**57.34%**	67.60%

Table 5.4: Human evaluation results: visual relevance. Rows show the selection accuracy for each category and their average. 'Real' stands for using original audios.

real audio.

Real or fake determination: In this task, we would like to see whether the generated audios can fool people into thinking that they are real. We provide instructions to the turkers that the audio of the current video might be either real (originally belonging to this video) or fake (synthesis by computers). The criteria of being fake can be bad synchronization or poor quality such as containing unpleasing noise. In addition to the generated results from our proposed methods, we also include videos with the original audio as a control. As an additional baseline, we also combine the video with a random real audio from the same category. This baseline is rather challenging as it uses real audios. Each evaluation is performed by 3 turkers and we aggregate the votes.

The percentages for the audios being rated as real are shown in Table 5.5 for all methods including the baseline (*Base*) and the real audio. *Seq* and *Flow* methods outperform the *Frame* method except for the printer category. Unsurprisingly, *Base* achieves decent results on categories that are insensitive to synchronization like Printer and Snoring, but much worse than our methods on categories sensitive to synchronization such as Dog and Drum. One of the reasons that turkers consider some of the real cases as fake is that a few original audios might include light background music or other noise which appears not fitting with the visual content.

	Frame	Seq	Flow	Base	Real
Dog	61.46%	**64.32%**	62.24%	54.69%	89.06%
Chainsaw	71.09%	73.96%	**76.56%**	68.23%	93.75%
Water flowing	70.83%	77.60%	**81.25%**	77.86%	87.50%
Rail transport	79.69%	**83.33%**	80.47%	74.74%	90.36%
Fireworks	76.04%	76.82%	**78.39%**	75.78%	94.01%
Printer	73.96%	73.44%	71.35%	**75.00%**	89.32%
Helicopter	71.61%	74.48%	78.13%	**78.39%**	91.67%
Snoring	67.71%	73.44%	73.18%	**77.08%**	90.63%
Drum	62.24%	64.58%	**70.83%**	59.64%	93.23%
Baby crying	57.29%	64.32%	61.20%	**69.27%**	94.79%
Average	68.69%	72.63%	**73.36%**	71.07%	91.43%

Table 5.5: Human evaluation results: real or fake task where people judge whether a video-audio pair is real or generated. Percentages indicate the frequency of a pair being judged as real.

5.4 Conclusion

In this chapter, we introduced the task of generating realistic sound from videos in the wild. We created a dataset for this purpose, sampled from the AudioSet collection, based on which we trained three different visual-to-sound deep network variants. We also provided qualitative, quantitative and subjective experiments to evaluate the models and the generated audio results. Evaluations show that over 70% of the generated sound from our models can fool humans into thinking that they are real.

Future directions include explicitly recognizing and reasoning about objects in the video during sound generation, and reasoning beyond the pixels and temporal duration of the input frames for more contextual generation.

CHAPTER 6

Conclusions

In this dissertation, we have discussed learning beyond-pixel mappings from different aspects. Specifically we formulate the problem as inferring the information of broader spacial (motion) and temporal domain as well as predicting other modalities from visual data. We also demonstrate that these mappings could be learned by deep convolutional neural networks from either self-recorded videos or large-scale data from Internet. To deal with the noisy characteristic of the Internet data, we apply careful annotation using crowd-sourcing platforms (AMT).

In order to learn the nature of the future prediction, we define the basic unit of the problem by handling sequence prediction between two video clips in Chapter 2. We show that the simple ordering prediction could be the backbone algorithm for the future prediction task (to predict what will happen next, it is necessary to have the knowledge of the ordering of the daily events first.). An ego-centric dataset of everyday-life-activity, named FPPA, has been collected to train sequence prediction models under both generalized (models have been trained using various individuals and scenes) and personalized settings (model have been trained using single person's data).

In Chapter 3, we explore object future state prediction by training a deep generative model to directly generate depictions of a specific object in it's future state. To enable the learning, we collect a novel time-lapse video dataset, which shows the entire or portion of object transformation process within a short period of time, from YouTube and we provide the annotations of the degree of the transformations. Meanwhile, we propose three tasks to model the future prediction under different scenarios.

In Chapter 4, we discuss a similar generation task but under the motion transfer scenario. Specifically, we propose a computational model to transfer any novel poses from a reference video to the target person. We train the personalized model using only a single YouTube video (several minutes long) of that person performing random activities. Then we could transfer any desired poses to the target person. This requires our model to generalize well to novel/unseen poses. We present a

two-stage framework to synthesize the foreground human and background scene separately, which achieves visually appealing generation results in a relatively high resolution.

Finally, in Chapter 5, we explore whether it is feasible to directly learn the mapping between visual input and the raw audio signals, given the observation that people perceive information from both visual and sound channels and these two modalities are corresponded. We propose several variations to encode the visual information and it has been jointly train with a sound generator to generate raw waveform samples from input video frames. We also collect a benchmark video-audio dataset for natural sound generation from the videos in the wild task. This dataset includes more than 28000 videos spanning 10 object categories.

6.1 What's next

Recently there have been many works exploring the relationship between visual and sound modalities. In the task of sound localization [Arandjelović and Zisserman, 2018, Senocak et al., 2018, Zhao et al., 2018], given the input image/video frames as well as the according sound, the model would like to predict the region in the image where the sound is most probably generated from. We expect to combine the localization technique with the sound generation tasks to generate/edit sound based on the visual manipulation. Specifically, when modifying the visual contents (such as removing/adding in some objects or moving an object closer/further), we expect the model could learn the differences and edit the generated sound accordingly.

Including our work [Zhou et al., 2019], there are an increasing number of works which are trying to learn the mapping between human poses and the photo-realistic video frames [Ma et al., 2017, Siarohin et al., 2018, Balakrishnan et al., 2018, Chan et al., 2018]. We would like to perceive the problem from a different aspect to learn the mapping between music/sound and human pose. Specifically, given a chunk of music, can we predict a sequence of reasonable human dancing poses which are consistent with the genre and beats of the music? Then we can see those mappings become a more complete story that we could use music to drive the movement of a real person in the picture.

Besides exploring more applications using generative models on visual or sound and making the generation more realistic and natural, the research topic from the opposite direction (distinguish whether a video/an audio is real or fake.) is also indispensable. With the devel-

opment of high performance generative models, discriminating deepfake[1] videos from the real ones on the video-sharing platforms or social media can be an increasingly important task. Recent works [Wang et al., 2019, Agarwal et al., 2019, Huh et al., 2018] distinguish fake images/video frames by learning the subtle attributes of human faces, personalized facial motion patterns or based on camera information (EXIF data). We are also interested in providing other concrete solutions to discriminate deepfake videos under a large-scale data setting as future works.

Back to the more fundamental problem, in our works, we utilize large-scale video data to train carefully designed networks and we show that the networks could learn reasonable mappings from one domain to the other. Now the question is can we say the networks understand what they have learned? Or say if the inputs are completely different types of data (e.g. including unseen objects) from the training data, can we still get reasonable feedback from the models? As future works, we would like to visualize the neurons of the generation task models to get some insights about what the network have learned. Also, we hope to reason the generative models such as combining cause and effect relationships or even more general knowledge graphs while doing the prediction to provide more insights about what the model has learned as well as to improve the generalization capability of the model.

[1]https://en.wikipedia.org/wiki/Deepfake

REFERENCES

[Abu-El-Haija et al., 2016] Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., and Vijayanarasimhan, S. (2016). Youtube-8m: A large-scale video classification benchmark. *CoRR*.

[Agarwal et al., 2019] Agarwal, S., Farid, H., Gu, Y., He, M., Nagano, K., and Li, H. (2019). Protecting world leaders against deep fakes. In *Computer Vision and Pattern Recognition (CVPR) Workshops*.

[Aghazadeh et al., 2011] Aghazadeh, O., Sullivan, J., and Carlsson, S. (2011). Novelty detection from an ego-centric perspective. In *Computer Vision and Pattern Recognition (CVPR)*.

[Alahi et al., 2016] Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., and Savarese, S. (2016). Social lstm: Human trajectory prediction in crowded spaces. In *Computer Vision and Pattern Recognition (CVPR)*.

[Arandjelović and Zisserman, 2018] Arandjelović, R. and Zisserman, A. (2018). Objects that sound. In *European Conference on Computer Vision (ECCV)*.

[Balakrishnan et al., 2018] Balakrishnan, G., Zhao, A., Dalca, A. V., Durand, F., and Guttag, J. (2018). Synthesizing images of humans in unseen poses. In *Computer Vision and Pattern Recognition (CVPR)*.

[Cao et al., 2017] Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. In *Computer Vision and Pattern Recognition (CVPR)*.

[Chan et al., 2018] Chan, C., Ginosar, S., Zhou, T., and Efros, A. A. (2018). Everybody Dance Now. *CoRR*.

[Chen et al., 2017] Chen, L., Srivastava, S., Duan, Z., and Xu, C. (2017). Deep cross-modal audio-visual generation. *CoRR*.

[Cho et al., 2014] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[Dekel Basha et al., 2012] Dekel Basha, T., Moses, Y., and Avidan, S. (2012). Photo sequencing. In *Computer Vision and Pattern Recognition (CVPR)*.

[Dekel Basha et al., 2013] Dekel Basha, T., Moses, Y., and Avidan, S. (2013). Space-time tradeoffs in photo sequencing. In *International Conference on Computer Vision (ICCV)*.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *Computer Vision and Pattern Recognition (CVPR)*.

[Donahue et al., 2013] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2013). Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*.

[Fang et al., 2018] Fang, H.-S., Lu, G., Fang, X., Xie, J., Tai, Y.-W., and Lu, C. (2018). Weakly and semi supervised human body part parsing via pose-guided knowledge transfer. *Computer Vision and Pattern Recognition (CVPR)*.

[Fang et al., 2017] Fang, H.-S., Xie, S., Tai, Y.-W., and Lu, C. (2017). RMPE: Regional multi-person pose estimation. In *International Conference on Computer Vision (ICCV)*.

[Fathi et al., 2011a] Fathi, A., Farhadi, A., and Rehg, J. M. (2011a). Understanding egocentric activities. In *International Conference on Computer Vision (ICCV)*.

[Fathi et al., 2012a] Fathi, A., Hodgins, J., and Rehg, J. (2012a). Social interactions: A first-person perspective. In *Computer Vision and Pattern Recognition (CVPR)*.

[Fathi et al., 2012b] Fathi, A., Li, Y., and Rehg, J. (2012b). Learning to recognize daily actions using gaze. In *European Conference on Computer Vision (ECCV)*.

[Fathi et al., 2011b] Fathi, A., Ren, X., and Rehg, J. (2011b). Learning to recognize objects in egocentric activities. In *Computer Vision and Pattern Recognition (CVPR)*.

[Gemmeke et al., 2017] Gemmeke, J. F., Ellis, D. P. W., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., and Ritter, M. (2017). Audio set: An ontology and human-labeled dataset for audio events. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

[Girshick et al., 2014] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)*.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*.

[Güler et al., 2018] Güler, R. A., Neverova, N., and Kokkinos, I. (2018). Densepose: Dense human pose estimation in the wild. In *Computer Vision and Pattern Recognition (CVPR)*.

[Gulrajani et al., 2017] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems (NIPS)*.

[Han et al., 2015] Han, X., Leung, T., Jia, Y., Sukthankar, R., and Berg, A. C. (2015). Matchnet: Unifying feature and metric learning for patch-based matching. In *Computer Vision and Pattern Recognition (CVPR)*.

[He et al., 2017] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *International Conference on Computer Vision (ICCV)*.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*

[Huh et al., 2018] Huh, M., Liu, A., Owens, A., and Efros, A. A. (2018). Fighting fake news: Image splice detection via learned self-consistency. In *European Conference on Computer Vision (ECCV)*.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*.

[Jaderberg et al., 2015] Jaderberg, M., Simonyan, K., Zisserman, A., and kavukcuoglu, k. (2015). Spatial transformer networks. In *Advances in Neural Information Processing Systems (NIPS)*.

[Jia et al., 2014] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R. B., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *CoRR*.

[Jolicoeur-Martineau, 2018] Jolicoeur-Martineau, A. (2018). The relativistic discriminator: a key element missing from standard GAN. *CoRR*.

[Karpathy et al., 2014] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR)*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*.

[Kitani et al., 2011] Kitani, K., Okabe, T., Sato, Y., and Sugimoto, A. (2011). Fast unsupervised ego-action learning for first-person sports videos. In *Computer Vision and Pattern Recognition (CVPR)*.

[Kitani et al., 2012] Kitani, K. M., Ziebart, B. D., Bagnell, J. A., and Hebert, M. (2012). Activity forecasting. In *European Conference on Computer Vision (ECCV)*.

[Lee et al., 2012] Lee, Y. J., Ghosh, J., and Grauman, K. (2012). Discovering important people and objects for egocentric video summarization. In *Computer Vision and Pattern Recognition (CVPR)*.

[Liang et al., 2019] Liang, J., Jiang, L., Niebles, J. C., Hauptmann, A., and Fei-Fei, L. (2019). Peeking into the future: Predicting future person activities and locations in videos. *CoRR*.

[Liang et al., 2017] Liang, X., Lee, L., Dai, W., and Xing, E. P. (2017). Dual motion gan for future-flow embedded video prediction. In *International Conference on Computer Vision (ICCV)*.

[Lin et al., 2017a] Lin, T.-Y., DollÃ¡r, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017a). Feature pyramid networks for object detection. In *Computer Vision and Pattern Recognition (CVPR)*.

[Lin et al., 2017b] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and DollÃ¡r, P. (2017b). Focal loss for dense object detection. In *International Conference on Computer Vision (ICCV)*.

[Liu et al., 2018] Liu, L., Xu, W., Zollhoefer, M., Kim, H., Bernard, F., Habermann, M., Wang, W., and Theobalt, C. (2018). Neural animation and reenactment of human actor videos. *CoRR*.

[Liu et al., 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*.

[Lu and Grauman, 2013] Lu, Z. and Grauman, K. (2013). Story-driven summarization for egocentric video. In *Computer Vision and Pattern Recognition (CVPR)*.

[Ma et al., 2017] Ma, L., Jia, X., Sun, Q., Schiele, B., Tuytelaars, T., and Van Gool, L. (2017). Pose guided person image generation. In *Advances in Neural Information Processing Systems (NIPS)*.

[Mao et al., 2017] Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. (2017). Least squares generative adversarial networks. In *International Conference on Computer Vision (ICCV)*.

[Mathieu et al., 2015] Mathieu, M., Couprie, C., and LeCun, Y. (2015). Deep multi-scale video prediction beyond mean square error. *CoRR*.

[Mehri et al., 2016] Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., Courville, A. C., and Bengio, Y. (2016). Samplernn: An unconditional end-to-end neural audio generation model. *International Conference on Learning Representations (ICLR)*.

[Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*.

[Neverova et al., 2018] Neverova, N., Alp Guler, R., and Kokkinos, I. (2018). Dense pose transfer. In *European Conference on Computer Vision (ECCV)*.

[Nie et al., 2018] Nie, X., Feng, J., and Yan, S. (2018). Mutual learning to adapt for joint human parsing and pose estimation. In *European Conference on Computer Vision (ECCV)*.

[Owens et al., 2016] Owens, A., Isola, P., McDermott, J., Torralba, A., Adelson, E., and Freeman, W. (2016). Visually indicated sounds. In *Computer Vision and Pattern Recognition (CVPR)*.

[Pathak et al., 2016] Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T., and Efros, A. (2016). Context encoders: Feature learning by inpainting. In *Computer Vision and Pattern Recognition (CVPR)*.

[Pickup et al., 2014a] Pickup, L. C., Pan, Z., Wei, D., Shih, Y., Zhang, C., Zisserman, A., Scholkopf, B., and Freeman, W. T. (2014a). Seeing the arrow of time. In *Computer Vision and Pattern Recognition (CVPR)*.

[Pickup et al., 2014b] Pickup, L. C., Pan, Z., Wei, D., Shih, Y., Zhang, C., Zisserman, A., Schölkopf, B., and Freeman, W. T. (2014b). Seeing the arrow of time. In *Computer Vision and Pattern Recognition (CVPR)*.

[Pirsiavash and Ramanan, 2012] Pirsiavash, H. and Ramanan, D. (2012). Detecting activities of daily living in first-person camera views. In *Computer Vision and Pattern Recognition (CVPR)*.

[Radford et al., 2015] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*.

[Ranzato et al., 2014] 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. *CoRR*.

[Redmon and Farhadi, 2017] Redmon, J. and Farhadi, A. (2017). Yolo9000: Better, faster, stronger. In *Computer Vision and Pattern Recognition (CVPR)*.

[Ryoo and Matthies, 2013] Ryoo, M. and Matthies, L. (2013). First-person activity recognition: What are they doing to me? In *Computer Vision and Pattern Recognition (CVPR)*.

[Sakoe and Chiba, 1978] Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

[Senocak et al., 2018] Senocak, A., Oh, T.-H., Kim, J., Yang, M.-H., and So Kweon, I. (2018). On learning association of sound source and visual scenes. In *Computer Vision and Pattern Recognition (CVPR)*.

[Siarohin et al., 2018] Siarohin, A., Sangineto, E., LathuiliÃ¨re, S., and Sebe, N. (2018). Deformable gans for pose-based human image generation. In *Computer Vision and Pattern Recognition (CVPR)*.

[Simon et al., 2017] Simon, T., Joo, H., Matthews, I., and Sheikh, Y. (2017). Hand keypoint detection in single images using multiview bootstrapping. In *Computer Vision and Pattern Recognition (CVPR)*.

[Simonyan and Zisserman, 2014a] Simonyan, K. and Zisserman, A. (2014a). Two-stream convolutional networks for action recognition in videos. *Advances in Neural Information Processing Systems (NIPS)*.

[Simonyan and Zisserman, 2014b] Simonyan, K. and Zisserman, A. (2014b). Very deep convolutional networks for large-scale image recognition. *CoRR*.

[Soomro et al., 2012] Soomro, K., Zamir, A. R., Shah, M., Soomro, K., Zamir, A. R., and Shah, M. (2012). Ucf101: A dataset of 101 human actions classes from videos in the wild. *CoRR*.

[Srivastava et al., 2015] Srivastava, N., Mansimov, E., and Salakhudinov, R. (2015). Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning (ICML)*.

[Sun et al., 2010] Sun, D., Roth, S., and Black, M. J. (2010). Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR)*.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems (NIPS).

[van den Oord et al., 2016] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *CoRR*.

[Villegas et al., 2017] Villegas, R., Yang, J., Zou, Y., Sohn, S., Lin, X., and Lee, H. (2017). Learning to generate long-term future via hierarchical prediction. *CoRR*.

[Walker et al., 2014] Walker, J., Gupta, A., and Hebert, M. (2014). Patch to the future: Unsupervised visual prediction. In *Computer Vision and Pattern Recognition (CVPR)*.

[Walker et al., 2017] Walker, J., Marino, K., Gupta, A., and Hebert, M. (2017). The pose knows: Video forecasting by generating pose futures. In *International Conference on Computer Vision (ICCV)*.

[Wang et al., 2019] Wang, S.-Y., Wang, O., Owens, A., Zhang, R., and Efros, A. A. (2019). Detecting photoshopped faces by scripting photoshop. In *CoRR*.

[Wang et al., 2018] Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Tao, A., Kautz, J., and Catanzaro, B. (2018). High-resolution image synthesis and semantic manipulation with conditional gans. In *Computer Vision and Pattern Recognition (CVPR)*.

[Xue et al., 2016] Xue, T., Wu, J., Bouman, K., and Freeman, B. (2016). Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*.

[Yuen and Torralba, 2010] Yuen, J. and Torralba, A. (2010). A data-driven approach for event prediction. In *European Conference on Computer Vision (ECCV)*.

[Zhang et al., 2017] Zhang, Z., Wu, J., Li, Q., Huang, Z., Traer, J., McDermott, J. H., Tenenbaum, J. B., and Freeman, W. T. (2017). Generative modeling of audible shapes for object perception. In *International Conference on Computer Vision (ICCV)*.

[Zhao et al., 2018] Zhao, H., Gan, C., Rouditchenko, A., Vondrick, C., McDermott, J., and Torralba, A. (2018). The sound of pixels. In *European Conference on Computer Vision (ECCV)*.

[Zhou et al., 2014] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014). Learning Deep Features for Scene Recognition using Places Database. *Advances in Neural Information Processing Systems (NIPS)*.

[Zhou and Berg, 2015] Zhou, Y. and Berg, T. L. (2015). Temporal perception and prediction in ego-centric video. In *International Conference on Computer Vision (ICCV)*.

[Zhou et al., 2019] Zhou, Y., Wang, Z., Fang, C., Bui, T., and Berg, T. L. (2019). Dance dance generation: Motion transfer for internet videos. *CORR*.