Human Motion Anticipation and Recognition from RGB-D

Emad Barsoum

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Engineering Science
in the Fu Foundation School of Engineering and Applied Science

COLUMBIA UNIVERSITY

2019

ABSTRACT

Human Motion Anticipation and Recognition from RGB-D

Emad Barsoum

Predicting and understanding the dynamic of human motion has many applications such as motion synthesis, augmented reality, security, education, reinforcement learning, autonomous vehicles, and many others. In this thesis, we create a novel end-to-end pipeline that can predict multiple future poses from the same input, and, in addition, can classify the entire sequence. Our focus is on the following two aspects of human motion understanding:

1. **Probabilistic human action prediction:** Given a sequence of human poses as input, we sample multiple possible future poses from the same input sequence using a new GAN-based network.
2. **Human motion understanding:** Given a sequence of human poses as input, we classify the actual action performed in the sequence and improve the classification performance using the presentation learned from the prediction network.

We also demonstrate how to improve model training from noisy labels, using facial expression recognition as an example. More specifically, we have 10 taggers to label each input image, and compare four different approaches: majority voting, multi-label learning, probabilistic label drawing, and cross-entropy loss. We show that the traditional majority voting scheme does not perform as well as the last two

approaches that fully leverage the label distribution. We shared the enhanced FER+ data set with multiple labels for each face image with the research community[1].

For predicting and understanding of human motion, we propose a novel sequence-to-sequence model trained with an improved version of generative adversarial networks (GAN). Our model, which we call HP-GAN2, learns a probability density function of future human poses conditioned on previous poses. It predicts multiple sequences of possible future human poses, each from the same input sequence but seeded with a different vector z drawn from a random distribution. Moreover, to quantify the quality of the non-deterministic predictions, we simultaneously train a motion-quality-assessment model that learns the probability that a given skeleton pose sequence is a real or fake human motion.

In order to classify the action performed in a video clip, we took two approaches. In the first approach, we train on a sequence of skeleton poses from scratch using random parameters initialization with the same network architecture used in the discriminator of the HP-GAN2 model. For the second approach, we use the discriminator of the HP-GAN2 network, extend it with an action classification branch, and fine tune the end-to-end model on the classification tasks, since the discriminator in HP-GAN2 learned to differentiate between fake and real human motion. So, our hypothesis is that if the discriminator network can differentiate between synthetic and real skeleton poses, then it also has learned some of the dynamics of a real human motion, and that those dynamics are useful in classification as well. We will show through multiple experiments that that is indeed the case.

[1]`https://github.com/Microsoft/FERPlus`

Therefore, our model learns to predict multiple future sequences of human poses from the same input sequence. We also show that the discriminator learns a general representation of human motion by using the learned features in an action recognition task. And we train a motion-quality-assessment network that measure the probability of a given sequence of poses are valid human poses or not.

We test our model on two of the largest human pose datasets: NTURGB-D, and Human3.6M. We train on both single and multiple action types. The predictive power of our model for motion estimation is demonstrated by generating multiple plausible futures from the same input and showing the effect of each of the several loss functions in the ablation study. We also show the advantage of switching to GAN from WGAN-GP, which we used in our previous work. Furthermore, we show that it takes less than half the number of epochs to train an activity recognition network by using the features learned from the discriminator.

Contents

List of Figures

Acronyms

Notation	Description
3DCNN	3D Convolution Neural Network
AE	Autoencoder
CNN	Convolution Neural Network
DCNN	Deep Convolutional Neural Network
FC	Fully Connected
fps	frames per second
GAN	Generative Adversary Network

Notation	**Description**
GDL	Gradient Difference Loss
GLOH	Gradient Location and Orientation Histogram
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HAM	Hierarchical Attention Memory
HMM	Hidden Markov Model
HOF	Histogram of Optical Flow
HOG	Histogram of oriented gradients
HP-GAN	Human Prediction via Generative Adversarial Network
LSTM	Long Short-Term Memory
MBH	Motion Boundary Histogram
NTM	Neural Turing Machine
P-LSTM	Part Long Short-Term Memory

Notation	**Description**
R-CNN	Region-Based Convolutional Neural Network
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SIFT	Scale-invariant feature transform
SLAM	Simultaneous Localization and Mapping
SR	Semantic Representation
SURF	Speeded up robust features
SVM	Support Vector Machine
SVO	Subject Verb Object
TEM	Temporal Modeler
VAE	Variational Autoencoder
WGAN	Wasserstein Generative Adversarial Network
WGAN-GP	Wasserstein Generative Adversarial Network - Gradient Penalty

Acknowledgements

There are a lot of people without their support and encouragement I would not have started or continued my doctoral degree study, especially while working. I am indebted to my family, friends and coworkers.

Firstly, I want to express my sincere gratitude to my advisor Prof. John Kender for putting up with me during all those years. And for all the help that he provided, his continuous support of my research, and all the fruitful discussions that we had on various research topics. I learned a lot from John expertise and gained confidence in my research. Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Shih-Fu Chang, Prof. Steven Feiner, Prof. Zoran Kostic and Prof. Carl Vondrick for reviewing the thesis and accepting being part of the committee.

Secondly, I would like to thank Tommer Leyvand, my manager at Microsoft when I applied to Columbia University, for encouraging me to pursue my degree and writing one of my recommendation letters, even knowing that my study might take time from my work. I also would like to thank Dr. Zicheng Liu and Dr. Cha Zhang from Microsoft. I learned a lot from them, our collaboration on various research topics and vision projects helped me in the field and improved my research experience especially with all the debates and open-ended discussions that I had with them.

Thirdly, I would like to thank both Columbia University and Microsoft. Columbia, for offering such a great program from a such prestigious university and Microsoft for offering financial help.

Last but not least, I would like to thank my family for supporting me, especially my wife Rania Canaan, during my course work, research and writing of this dissertation. And my kids Christopher and Clara, who do not understand yet why I spend a little time with them.

To my kids, Christopher and Clara Barsoum

1

Introduction

Understanding time-based visual contents and anticipating the short-term future is a fundamental human trait. Such abilities are vital to how we interact with the world and how we summarize the visual information around us. With the proliferation of multimedia data captured from social apps, phones, action cameras, security systems, depth sensors and many other devices, it becomes critical to be able to mimic such human skill and automate the process of understanding, annotating, and predicting visual contents.

In this thesis we develop a novel end-to-end pipeline that can predict and classify visual time based input from RGB-D and skeleton data. The primary focus is to understand the dynamics of human action within these RGB-D segments, in order to be able to predict and classify short-term future actions.

1.1 Activity prediction

Accurate short-term (several seconds) human-scale predictions of what will happen in the world given past events is a fundamental and useful human ability. Such aptitude is vital for daily activities, social interactions, and ultimately survival. For example, driving requires predicting other cars' and pedestrians' motions in order to avoid an accident; handshaking requires predicting the location of the other person's hand; and playing sports requires predicting other players' reactions. In order to create a machine that can interact seamlessly with the world, it needs a similar ability to understand the dynamics of the human world, and to predict probable futures based on learned history and the immediate present.

However, the future is not deterministic, so predicting the future cannot be deterministic, except in the very short term. As the predictions extend further into the future, uncertainty becomes higher. People walking may turn or fall; people throwing a ball may drop it instead. Nevertheless, some predictions are more plausible than others, and have a higher probability to occur.

In this thesis, we focus on creating a novel model that can predict multiple plausible future human (skeleton) poses from a given past. The number of poses taken from the immediate past, and the predicted number of poses in the future–which can be unrestricted–are parameters to the model. To accomplish this, we use a modified version of the improved Wasserstein generative adversarial network (WGAN-GP) Gulrajani et al. 2017 with a custom loss function that takes into consideration specifics human motion and human anatomy in Barsoum, Kender, and Liu 2018. And we improved our approach from ibid. with a new GAN algorithm that combine some aspect of Wasserstein Generative Adversarial Network - Gradient Penalty (WGAN-GP) and the original GAN in order to have a discriminator that compute a probability between real and fake human poses, loss that indicate the quality of the prediction and a more stable trainer. We also added a way to control the diversity of the generated poses and an energy loss inspired from human motion animation.

The generator is a novel adaptation of a sequence-to-sequence model Sutskever, Vinyals, and Le 2014 of poses derived from a Recurrent Neural Network (RNN), the discriminator is a multi-layer network (MLP) with a skip connection and the quality assessment network is a recurrent network with temporal attention. We use the discriminator network to train the generator and the quality assessment network to learn how to distinguish a real sequence of poses from a fake one. In essence, we propose and investigate a novel combination of some aspects of the original GAN Goodfellow et al. 2014 with WGAN-GP Gulrajani et al. 2017. Additionally, we train our model on all actions at once, so its output is not conditioned on any specific human action. Our model takes as input a sequence of previous skeletal poses, plus a random vector z from a uniform distribution, to create a sequence of possible future poses. For each such z value, the model generates a different output sequence. Therefore, the value z sample multiple possible future from the input sequence.

We use an Recurrent Neural Network (RNN) for the generator because RNNs are a class of neural networks designed to model sequences, especially variable length sequences. They have been successfully used in machine translation Sutskever, Vinyals, and Le 2014, caption generation from images Donahue et al. 2015; Jia et al. 2015, video classification and action recognition Baccouche et al. 2011; Donahue et al. 2015; Ng et al. 2015a, action detection Yeung et al. 2016, video description Donahue et al. 2015; Kiros, Salakhutdinov, and Zemel 2014; Pan et al. 2016; Yao et al. 2015b, sequence prediction Graves 2013; Srivastava, Mansimov, and Salakhutdinov 2015 and others.

We structure the learning by using GAN algorithm because it allows us to generate future poses that are not tied to specific ground truth, by that we mean that the loss used by the generator to learn how to generate realistic poses does not include the actual ground truth of the dataset. GANs Goodfellow et al. 2014 are a class of unsupervised learning algorithms, inspired by minmax game theory Sion 1958, it is based on two networks that compete against each other. A generator network tries to create random samples that are indistinguishable from training samples, and a discriminator (or critic) network tries to distinguish the generated sample from the real one.

GANs have weaknesses. They can be difficult to train and unstable in their learning, their loss value does not necessarily indicate the quality of the generated sample, and the training can collapse easily. Recent literature Arjovsky, Chintala, and Bottou 2017; Gulrajani et al. 2017; Mao et al. 2016; Neyshabur, Bhojanapalli, and Chakrabarti 2017; Qi 2017; Uehara et al. 2016 tries to improve GAN training and provide a theoretical guaranty for its convergence. In our work, we address this by adding a custom loss based on skeleton physics in addition to the GAN loss, in order to stabilize and improve the training.

Finally, to quantitatively assess the quality of the non-deterministic predictions, we simultaneously train a motion quality assessment model that learns the probability that a given skeleton sequence is a real human motion.

We test our motion prediction model on two large datasets each captured with a different modality. The first is the NTURGB-D Shahroudy et al. 2016a dataset, which is the largest available RGB-D and skeleton-based dataset, with data captured by Microsoft Kinect v2 sensor. The second is the Human3.6M Ionescu, Li, and Sminchisescu 2011; Ionescu et al. 2014 dataset, which is one of the largest available datasets derived from motion capture (MoCap) data.

1.2 Video understanding

The high level architecture of video understanding pipeline is usually composed of three phases:

1. Extract multiple features from video, either per-frame features or per-multiple-frames features (i.e., 3D CNN).
2. Combine them into a descriptor, such as a fixed length vector through pooling or other statistical methods, or a variable length vector that takes advantage of the temporal dynamics, such as RNN and its family of networks.
3. Classify the features, which is usually done by Support Vector Machine (SVM) or softmax.

Most recent work on video understanding combine Convolution Neural Network (CNN) for per-frame feature extraction, with a variant of RNN or some polling mechanism, for temporal analysis and classification. Some work uses CNN per-frame, and fuses them with 3D convolution and pooling, followed by a softmax for classification. Using CNN as per-frame color feature extractor to the next computing unit, in the video understanding pipeline, can also be extended to work with depth frames in the same way as color frames, and to take advantage of existing pretraining models for color images. The output of this phase is a sequence of features per clip that can be aggregated with RNN using an attention model.

Since its introduction in machine translation Bahdanau, Cho, and Bengio 2014, attention mechanisms have gained traction in the vision community Xu et al. 2015. In this thesis, we use temporal attention in the quality assessment network to take all past poses into consideration for deciding if the input sequence is valid or not. We previously have used this approach in emotion recognition from an audio stream Mirsamadi, Barsoum, and Zhang 2017, which detected those temporal regions in the audio stream that exhibited emotion.

Although researchers now have more labeled video and depth data than ever before, still the amount of unlabeled data is far larger. There has been much interest in unsupervised learning in order to take advantage of some of the many available unlabeled datasets. Most unsupervised learning works by defining an energy function whose value is low on the observed data and high otherwise. But notably, GAN Goodfellow et al. 2014 has shown impressive results by removing the need to design an energy function. In the context of video, GAN has been used to predict future frames Mathieu, Couprie, and LeCun 2015. Such an ability demonstrates that it is a powerful mechanism to learn the temporal dynamics in the video clip. This thesis applies a similar mechanism on skeleton data, which is a novel contribution. The ability to predict an action before it happens gives us the flexibility to make decisions about human motion much faster.

1.3 Motivation

Video recognition and prediction is a hard problem and still an open research area; solving it will enable a lot of important applications. Here a partial subset of applications that can be enabled with a robust video understanding algorithm.

- **Security:** A growing number of cities now have cameras nearly everywhere for security. With such a large number of cameras, it is impractical for humans to monitor them, which reduces their benefit. Having a service that continuously monitors those streams and notifies authorities for any suspicious activities is extremely important.
- **Safety:** Many situations call for safety monitoring, such as caring for infants, guarding swimming pools, watching over the elderly, and securing farm animals. In these cases, we need to understand in real-time if the actual action or pose is a safety risk or not, and act accordingly.
- **Learning:** Often, education consists in providing immediate feedback to students. Whether learning martial arts, dancing, or any activity that requires a timed pose, it would be ideal to be able to understand the current activity and compare it to a reference activity.
- **Robotics:** Currently, robots use Simultaneous Localization and Mapping (SLAM) to navigate. But to also recognize scenes in real-time would improve the autonomy of the robot and safety of people around the robot.
- **Augmented Reality (AR):** With the introduction of Microsoft HoloLens and similar systems, there has been an increase excitement about AR and its

applications. One of the more important applications is scene understanding, with continuous visual and semantic feedback to the user.

- **Self-driving car:** Besides understanding the scene, a self-driving car needs to differentiate between pedestrians and non-pedestrians. This is a critical first step to reduce the probability of fatal accidents.
- **Searching:** There is a plethora of video contents uploaded on the web from multitude of sources. To search for archived video contents, systems are needed to recognize the contents of those videos and to index them for fast access.
- **Gesture:** With the increased focus on Virtual Reality (VR) and AR, there is a need to interact with the virtual worlds naturally. One of the most natural ways is to use hand gestures similar to the way we interact with the physical world.

The above are a subset of potential applications which depend on a strong video understanding pipeline in order to function reliably. The motivation of our work is that we find the challenges of video recognition agreeable, and their potential applications valuable. The recognition and prediction of human motions should make life better and safer.

1.4 Contributions

As our main contributions, we have developed the following:

1. A novel human motion model which predicts multiple possible futures from a single past, and which classifies that human activity.

2. A new loss function which controls the diversity of the predicted poses.
3. A motion quality assessment model which quantitatively evaluates the quality of the predicted human motions.
4. Improved stability of GAN training for non-Wasserstein Generative Adversarial Network (WGAN), by using some aspects of WGAN-GP for non-WGAN.

1.5 Outline of the dissertation

The rest of the thesis is organized as follows. Related work on both prediction and recognition are discussed next, in Chapter 2. Then, we discuss various classification techniques and how to improve them, in Chapter 3. We follow with discussion on prediction and unsupervised learning, including the latest trends, in Chapter 4. Having discussed the state of the art and its strengths and weaknesses, we then present an overview of the two RGB-D dataset used in our research, NTURGB-D and Human3.6m, in Chapter 5. We then present our algorithms and results for probabilistic human motion prediction and classification, in Chapter 6. We measure and compare its effectiveness under different experimental circumstances and ablation studies, in Chapter 7. We conclude with a summary and an outline of future work, in Chapter 8.

2 Related work

Given the availability of large video datasets and affordable 3D sensors, there has been a lot interest in human motion classification and prediction. Classification research usually focuses on multi-modality, by combining color, depth, IR, and skeleton data streams. Prediction research usually focuses on the prediction of future skeleton data. (We note that there is some work on predicting future frames in the pixel space, but those results are very limited).

2.1 Motion prediction

The ability to predict human poses conditioned on previous poses using deep RNNs Fragkiadaki et al. 2015; Jain et al. 2016; Martinez, Black, and Romero 2017 is due in part to the availability of large human motion datasets Ionescu, Li, and Sminchisescu 2011; Ionescu et al. 2014; Shahroudy et al. 2016a. In this section we review prior work on prediction, and indicate how our approach is a contribution to it.

In general, such motion prediction can be categorized into two categories: probabilistic and deterministic prediction.

2.1.1 Probabilistic motion prediction

Most work in probabilistic human motion prediction uses non-deep learning approaches Fragkiadaki et al. 2015; Koppula and Saxena 2013; Lehrmann, Gehler, and Nowozin 2014; Pavlovic, Rehg, and MacCormick 2000; Sidenbladh, Black, and Sigal 2002; Wang, Fleet, and Hertzmann 2005, 2008. In Lehrmann, Gehler, and Nowozin 2014, the authors propose simple Markov models that model observed data, and use the proposed model for action recognition and task completion. The limitation in this approach is that it predicts motion from a single action only, and assumes that human motion satisfies the Markov assumption. In Wang, Fleet, and Hertzmann 2008, the authors introduce Gaussian process dynamical models (GPDMs) to model human pose and motion. However, they train their model on each action separately, and constrain the model to a Gaussian process. In Sidenbladh, Black, and Sigal 2002, the authors map human motion to a low dimensional space, and use the coordinates as an index into a binary tree that predicts the next pose, thus casting the prediction problem into a search problem. However, this approach can not generalize to previously unseen motions. In Pavlovic, Rehg, and MacCormick 2000, the authors use switching linear dynamic systems learned through a Bayesian network, and in Koppula and Saxena 2013 the author used conditional random fields (CRF) to model spatio-temporal dynamics.

In contrast to all the above, our work does not use any statistical models to constrain the motion. As far as we know, we are the first to use deep neural networks for probabilistic human motion prediction.

2.1.2 Deterministic motion prediction

Recent human motion prediction, which relies on deep RNNs Fragkiadaki et al. 2015; Jain et al. 2016; Martinez, Black, and Romero 2017 or deep neutral networks Bütepage et al. 2017, is primarily deterministic. In Fragkiadaki et al. 2015, the authors mix both deterministic and probabilistic human motion predictions. Their deterministic aspect is based on a modified RNN called Recurrent-Decoder (ERD) that adds fully connected layers before and after an LSTM Hochreiter and Schmidhuber 1997 layer, and minimizes an Euclidean loss. Their probabilistic aspect uses a Gaussian Mixture Model (GMM) with five mixture components, and minimizes the GMM negative log-likelihood. For both aspects, they predict a single future human pose at a single time step. To predict more, they recurrently feed the single predicted pose back to the input. One drawback of this approach is error drifting, where the prediction error of the current pose will propagate into the next pose. In contrast, we predict multiple human poses at once thus avoiding error drifting. In addition, we do not impose any statistical model constrains like GMM over the motions.

In Jain et al. 2016, the authors develop a general framework that converts a structure graph to an RNN, called a Structure-RNN (S-RNN). They test their framework on different problem sets including human motion prediction, and show that it outperforms the current state of the art. However, they need to design the structure graph manually and task-specifically. In Martinez, Black, and Romero 2017, the authors examine recent deep RNN methods for human motion prediction, and show that they achieve start-of-the-art results with a simpler model by proposing three simple changes to RNN. In Bütepage et al. 2017, the authors use an encoder-decoder network based on a feed-forward network, and compare the results from three different such architectures: symmetric, time-scale, and hierarchical.

However, the main issues of deterministic prediction of human motion are two-fold. The future is not deterministic, so the same previous poses could lead to multiple possible poses. And, their use of an L_2 norm can cause the model to average between two possible futures Mathieu, Couprie, and LeCun 2015, resulting in blurred motions.

2.1.3 Non-human motion prediction

The prediction of multiple possible futures using RNNs has several precedents. In Graves 2013, the authors use an LSTMHochreiter and Schmidhuber 1997 to generate text and handwriting from an input sequence. They generate one item at a time, by sampling the resultant probability. Then, they append the predicted item to the input sequence and remove its first item, and iterate. This creates a sequence of desired length, but the method may eventually create an input to the LSTM that does not contain any of the original input. In contrast, we pursue a method that trains our model to generate the entire desired output sequence of poses all at once.

The prediction of single or multiple possible futures using GANs also has precedents. InMathieu, Couprie, and LeCun 2015, the authors trained a convolution model for both the generator and the discriminator in order to predict future frames. They improved the predictions by adding an image gradient difference loss to the adversary loss. However, they again only predict a single possible future and, due to the use of the convolution network, the model can only predict a fixed length output. In contrast, we support variable length input and variable length output, and can also generate multiple possible futures from the same input.

In Chen et al. 2017, the authors predict or imagine multiple frames from a single image. They generate affine transformations between each frame, and apply them to the original input image to produce their prediction. Although this can imagine multiple futures from the same input image, a single image is not sufficient to capture the temporal dynamics of a scene. Furthermore, it makes the oversimplified assumption that the change between the images can be captured by using an affine transform.

2.2 Action recognition

There have been a number of researches on action recognition and video understanding. Until recently most of the publications start with hand-crafted features tailored to video data Chen and Hauptmann 2009; Kim, Wong, and Cipolla 2007; Laptev et al. 2008; Liu and Shah 2008; Sadanand and Corso 2012; Wang et al. 2011, such as a combination of Harris 3D Detector Laptev et al. 2008, Motion Boundary Histogram (MBH), Histogram of Optical Flow (HOF) and/or Dense Trajectories. Those features get combined into a descriptor that describe the entire clip. The descriptor is fed into a classifier, usually a SVM, in order to classify each clip. For action recognition on depth or skeleton data, most research uses a normalized version of skeleton joints as a feature, and feed this to Hidden Markov Model (HMM) or RNN in order to model the temporal dynamics of the action.

Since the breakthrough in classification performance using CNN in the Imagenet challenge in 2012 Krizhevsky, Sutskever, and Hinton 2012 and subsequent years He et al. 2015; Simonyan and Zisserman 2014b; Szegedy et al. 2015, CNN usage has replaced most hand-crafted features in almost all vision-based tasks. The focus now is to augment the architecture of deep networks with domain specific knowledge, while taking advantage of existing pre-trained models.

Deep learning has taken over most image-based vision algorithms and has achieved much better state-of-the-art compared to classical approaches. However, in video understanding, the move from hand-crafted features into a more data-driven approach using deep learning has occurred more gradually. One reason for this slow adoption has been three-fold: the lack of sizable labeled video data until recently, the complexity required to recognize the visual contents inside a video, and the needed computational resources.

Some of the early works on using deep learning for action recognition are Ji et al. 2013 and Baccouche et al. 2011. Ji et al. 2013 uses hand crafted features fed to a 3D CNN. It was one of the first papers that expands CNN to the temporal domain. Baccouche et al. 2011 is one of the first papers that uses 3D CNN for features and Long Short-Term Memory (LSTM) for temporal dynamics. They first run a person detector, then crop each frame around the person and subsample each frame to reduce memory consumption. Howeve, none of these early approaches could beat the state-of-the-art handcrafted approach for video recognition.

One of the first deep learning approach that matched classical methods was Simonyan and Zisserman 2014a. They use two separate CNN streams, one for color data and another for optical flow. They were inspired by human vision system that also uses two streams: the ventral stream to recognize color, shape, and category, and the dorsal stream to recognize motion, depth, and position. Karpathy et al. 2014 introduced the YouTube1M dataset, which was the largest action dataset until very recently. In ibid. they used multiple fusion mechanisms across randomly selected frames, in order to show the different possible approaches to classify video with CNN; surprisingly, they did not use RNN.

Ng et al. 2015b did a detailed comparison between using LSTM versus different pooling mechanism on top of a per-frame CNN using both color frame and optical flow. One of the surprising results was that pooling mechanisms, which ignore temporal information, matched the LSTM result. This result corroborates the results of our entry in a video based emotion competition, Bargal et al. 2016, in which we won third place out of 21 participants. In that work, we used statistics on the top of CNN outputs to create a per-clip descriptor, then fed those descriptors into an SVM classifier. Another interesting pooling mechanism that takes order into consideration is Fernando and Gould 2016 which implements a pooling algorithm called rank pooling. It provides different results depending on the order of the input sequence, since different sequence orders produce different descriptors.

Lately there are a number of attempts to incorporate reinforcement learning with deep learning, in order to select parts of the clip that are important for the target category. In Yeung et al. 2016 they used reinforcement learning in order to read selected glimpses from a video clip. Their network predicts the next position to be read from, instead of sampling or blindly reading the entire clip.

As for depth data, Shahroudy et al. 2016b introduced a modified version of LSTM called Part Long Short-Term Memory (P-LSTM) that groups related human joints together, but they only use skeleton joints as input. However, in addition, they introduced the largest available action-based depth and skeleton data to date.

2.3 Human motion quality assessment

Compared to the amount of work on motion editing and synthesis in computer graphics, the research on automatic motion quality evaluation has received little attention Harrison, Rensink, and Panne 2004; Hodgins, O'Brien, and Tumblin 1998; Reitsma and Pollard 2003; Ren et al. 2005. The existing techniques were typically designed for special types of motions, or for motions obtained from software used to edit character motion Wang, Chen, and Wang 2014. A novel aspect of our motion quality assessment model is that it is trained simultaneously together with the motion prediction model.

3

The Video Understanding Pipeline

In this chapter, we focus explicitly on four critical areas of the video understanding pipeline that are the most relevant to this thesis: object and activity recognition, ground truth labeling, feature extraction, and temporal dynamics modeling. For each of these areas, we summarize and critique the state of the art, then contrast it with the contributions that follow in this thesis.

There are multiple steps in the video recognition and description pipeline, as shown in Figure 3.1.

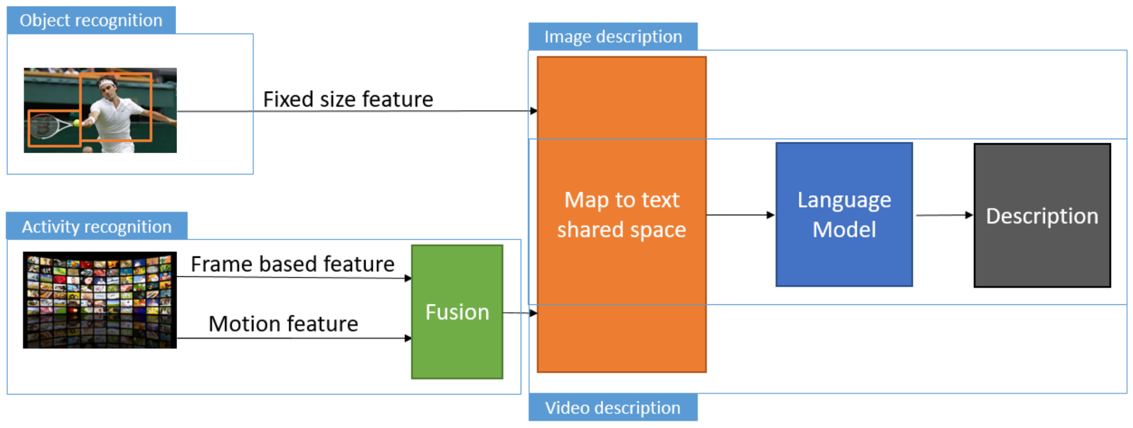


Figure 3.1: Video and image understanding pipeline.

Recognition starts with feature extraction, which can be done on a per-frame or per-sequence basis. Features can be extracted directly from raw pixel values through CNN or manually through computer vision algorithms, and concurrently from a pre-processed stream such as optical flow or skeleton poses in order to take temporal dynamics into consideration.

Next, per-frame features are combined, using one of two approaches: pooling, which ignores temporal information and computes some statistics from the given sequence of features, or temporal fusion, which uses RNN or its variants, or 3D Convolution Neural Network (3DCNN), in order to learn the temporal dynamics directly. The outputs of the above approaches are fed to a classifier, usually softmax or SVM. Then the output can be fed to a language model in the embedding space, in order to generate a detailed description.

We now explore some specific implemented variations of this basic pipeline.

3.1 Object and activity recognition

3.1.1 Object recognition methods

The ImageNet competition is one of the largest competitions on object recognition from still images. In recent years, the best performer reached more than 99% accuracy for top-5 classification. Comparing the winning architectures from the last couple of years, the main differences beyond minor architecture changes are: the use of deeper networks, longer training, and the ensembling the results of multiple different networks. Current research trends are focusing on more challenging tasks, such as object detection and segmentation, instant segmentation, and video understanding.

For video understanding, the per-frame features are usually extracted from one of the top models already pretrained on ImageNet data. Depending on the architecture, image features are extracted from the last fully connected layer or the last convolution layer, in order to preserve spatial information. For depth data, there are multiple methods, such as the following.

3.1.1.1 Using an RGB pretrained model

There are more separate color channel images (R, G, B) than there are depth images (D). Converting the depth frame to look like a color image, then feeding it to a pretrained RGB model, is one way we can take advantage of existing state of the art models. Our research has in fact used this approach with success, by converting a depth frame into a gray level image, then stacking three copies of this image together as red, green, and blue channels, respectively, to form a ”color image” compatible with the pretrained model. We then fed these resulting color images to several of the top pretrained ImageNet models such as VGGnet, ResNet, and Inception.

3.1.1.2 Training from scratch

Recently there has been interest in capturing large dataset of human activities using depth sensors. Two of the largest available RGB-D datasets are Ionescu et al. 2014 and Shahroudy et al. 2016b. With such large datasets, that contains color, depth and skeleton, we can train from scratch and explore various algorithms.

3.1.1.3 Manually defining the features

We can also extract manually customized vision features from depth frames and use those features for training. We can easily extend existing RGB feature such as Scale-invariant feature transform (SIFT), Histogram of oriented gradients (HOG), Speeded up robust features (SURF), Gradient Location and Orientation Histogram (GLOH) and many others. But, since a depth frame allow us to easily separate the foreground from the background, this helps segment the moving object in the scene, allowing us to ignore everything else.

3.1.2 Activity recognition methods

Until recently, most activity recognition algorithms depend heavily on a hybrid approach, mixing hand crafted methods with deep learning Karpathy et al. 2014. The main reasons were the lack of a large labeled video dataset, and the huge computational requirements. So, training end-to-end from scratch was not practical in most cases. In this subsection, we review some of the recent work done in video understanding and detection, and how it has progressed from a hybrid approach to a full deep learning approach.

3.1.2.1 Early work

One of the first researches that applied 3DCNN on a human action recognition task is the work done by Ji et al. 2013 in 2010. They extracted hand crafted features from each frame, then fed the result into 3DCNN. They did not use RNN or any of its variants for temporal information; they simply used the output of the 3DCNN as the classification output. Consequently, their approach ignored the order of the frames in the video. Their main contribution was that they were the first paper to extend a CNN kernel to the time domain, taking advantage of local temporal information with very little hand-crafted feature usage. But, in order to support longer temporal window, they could have used a RNN. Another drawback is that their first layer did in fact use some hand-crafted features that were not trained from the input data.

It appears that Baccouche et al. 2011 was the first paper to combine a 3DCNN with a LSTM for action recognition. They used a 3DCNN to extract spatio-temporal features from each clip, and a LSTM to classify the clip while taking frame order into consideration. The input to the 3DCNN was a cropped image from the output of a human figure detector. Possible improvements would have been to take advantage of the entire frame instead of its crop around a person, and to consider all the output of the sequence in the final classification.

3.1.2.2 Breakthrough work

While Baccouche et al. 2011; Ji et al. 2013 were ahead of their time, they did not match the best hand-crafted method in performance. The first deep learning approach that did was Simonyan and Zisserman 2014a in 2014. They trained two different streams, a spatial stream and a temporal stream, inspired by the human vision system. They implemented the "two-streams hypothesis" of Goodale and Milner 1992: a ventral stream for recognition, and a dorsal stream for motion.

The spatial stream was a standard CNN that took a randomly picked frame as input. The temporal stream extracted a hand-crafted feature from optical flow. They fused the output of both streams using a SVM and averaginf. Their main contribution was separating activity recognition into streams of motion and appearance, which beat state-of-the-art hand-crafted methods, by using end-to-end learning. Possible improvements would have been to train both streams jointly, to consider more than one frame for the appearance stream, and to learn the temporal dynamics from the training data.

3.1.2.3 Large dataset work

Most of the work mentioned so far used classical vision algorithms in addition to neural networks, due to the limited size of the data. Karpathy et al. 2014 introduced the first large scale dataset for action recognition, containing 1 million YouTube videos and 487 classes. All videos were weakly labeled, based on the tags associated with the video. This dataset was large enough for an end-to-end deep learning training. However, the remaining issue was the required computational resources.

Beside the introduction of the large dataset, ibid. compared different CNN architectures, from single frame input to multiple frame inpur, and with several different but very basic fusion techniques. The main contribution of this work was the large activity dataset itself. A possible follow up would have been to use a 3DCNN in conjunction with RNN, in order to support long sequences, and to add some aspects of visual attention, in order to take into consideration the spatial information of the frames.

Following up the release of this large dataset is the work of Ng et al. 2015b. They did a more comprehensive architecture comparison, with different pooling techniques and a LSTM, while incorporating features from optical flow in addition to per-frame features. The main highlight was an in-depth comparison among different techniques for classifying a long video sequence, and a comparison between pooling versus LSTM. A further follow up would be to compare different sampling rates, instead of their single 1 frames per second (fps).

3.1.2.4 Localization work

So far we have reviewed only video classification from an already (manually) trimmed video. A more challenging task is simultaneous action detection and localization. That is the ability to detect and locate an action within a long untrimmed clip, in addition to classifying the actual action or actions, since an untrimmed clip might have one or more actions. Recent research in action detection is the work done by Shou, Wang, and Chang 2016 and Yeung et al. 2016.

In Shou, Wang, and Chang 2016, they use a divide-and-conquer technique, by splitting the problem into three dependent problems, and creating a CNN for each of the sub-problems. The first network is the proposal network, which identifies a candidate segment from a long video that might contain an action. The second network is a classification network using one-vs-all classification. The third network is the localization network, which fine tunes the output of the classification network, using a custom loss function to improve the overlap with the ground truth.

In Yeung et al. 2016, they used a CNN to extract per-frame features, and a RNN to predict the next frame to look at and classify, plus reinforcement learning to accept or refuse the current candidate solution produced by the RNN layer. One of the main advantages of this approach was that the network learned during training where to look next, and therfore did not need to inspect each frame.

3.1.3 Video description methods

In video description, the goal is to generate a valid natural language description for the visual content of the video. This goal requires the computer to understand the activity happening in the video through time, and to use the language model in order to describe the activity coherently. Generating a description from a video requires combining video understanding and natural language understanding methods.

3.1.3.1 Early work

Similarly to video understanding, most research on video description until recently was mainly based on hand-crafted features. Rohrbach et al. 2013 was one of the first papers that used a machine translation technique in video description. They mapped a visual presentation to a Semantic Representation (SR), then mapped the SR to the target language. This was similar to an Encoder-Decoder model, and was inspired by existing machine translation models. Their main highlight was treating the video description problem as a language translation problem. A possible improvement would have been to replace the intermediate presentation SR with a more flexible presentation that takes temporal information into consideration.

About the same time, Guadarrama et al. 2013 introduced YouTube2Text, which is one of the largest datasets for video description. In their paper they extracted dense trajectories from the video, and ran object detection at 2 fps. They stacked both the motion and object features, and passed the result to a nonlinear SVM for training. As for their language model, they used semantic hierarchies learned from data, and a priori probabilities learned from the web. Their main highlight was the learning of Subject Verb Object (SVO) triples from a web-scale corpus, and their ability to work on out-of-domain actions–and additionally, their providing of a large annotated dataset for researchers. A potential improvement would have been to replace SVO triples with a much righer language model.

A bit later, Donahue et al. 2015 showed different ways of connecting a CNN to a LSTM, in order to solve the several different vision problems of activity recognition, image description, and video description. They used various combinations of CNN and LSTM. In their many-to-one system for activities recognition, the inputs were variable size sequences of frames and the output was a single classification for the clip. In their one-to-many system used in image description or image caption generation, the input is a single image and the output is a sequence of words. In their many-to-many system used for video description or video caption, the input is a sequence of frames and the output is a sequence of words that describe those frames. Most recent research in video description and video understanding are variations of these general architectures.

3.1.3.2 Attention work

One of the first papers to use an attention model in video description is Yao et al. 2015a. They use temporal attention, and show that the corresponding generated word represents what the model attends to. For each generated word, the temporal attention attends to the subset of the frames corresponding to those words, and therefore, the network learned to focus on the actual activities during training. They use a 3DCNN for high level feature extraction; however, the input to their 3DCNN network is a hand-crafted feature. The highlight of this paper is the use of a soft attention model across the time domain, but they needed hand-crafted features as input to their network.

Current research trends in generating descriptions from video data focus on multiple fronts. Some memorize a large presentation of the video data by adding auxiliary memory in order to have a global context of the entire video. Some look at different representation levels Ballas et al. 2015, in order to capture high-level and low-level feature sets. Some improve video processing performance in order to reduce computation resource. Some add an attention model in order to mimic deliberately how humans process video information.

3.1.4 Summary

In this section, we reviewed some of the milestone systems and building blocks relevant to the video recognition and description problem, together with the sharp transition from hand-crafted features to end-to-end learning pipelines. In this thesis, we focus on skeleton poses, and most of the recognition techniques mentioned here are relevant. But we also must address the challenge of generating a good label for video activity data. And, we still are faced with a relative lack of high quality ground truth for this task.

3.2 Ground truth labeling

One of the challenges for video recognition is getting good quality ground truth. For datasets that are large and diverse, label variation between taggers is high.

There are multiple ways to improve tagging in case of uncertainty. In this section we will use the better-studied problem of facial emotion recognition as a paradigm of these ways. Emotion based on appearance is subjective, and it is challenge to get good ground truth for emotion based on facial expression. However, these datasets are small enough to try multiple experiments, and we review several of them.

3.2.1 The FER+ data set

We start with the Facial Expression Recognition (FER) dataset. The original FER was prepared by Pierre Luc Carrier and Aaron Courville by web crawling face images with emotion related keywords. The images are filtered by human labelers, but the label accuracy is not very high Goodfellow et al. 2013. A few examples are given in Figure 3.2.

Therefore, in our prior work Bargal et al. 2016; Barsoum et al. 2016, we decided to re-tag the FER data set with crowd sourcing. For each input image, we asked crowd taggers to label the image into one of 8 emotion types: neutral, happiness, surprise, sadness, anger, disgust, fear, and contempt. The taggers were required to choose one single emotion for each image, and the gold standard method was adopted to ensure

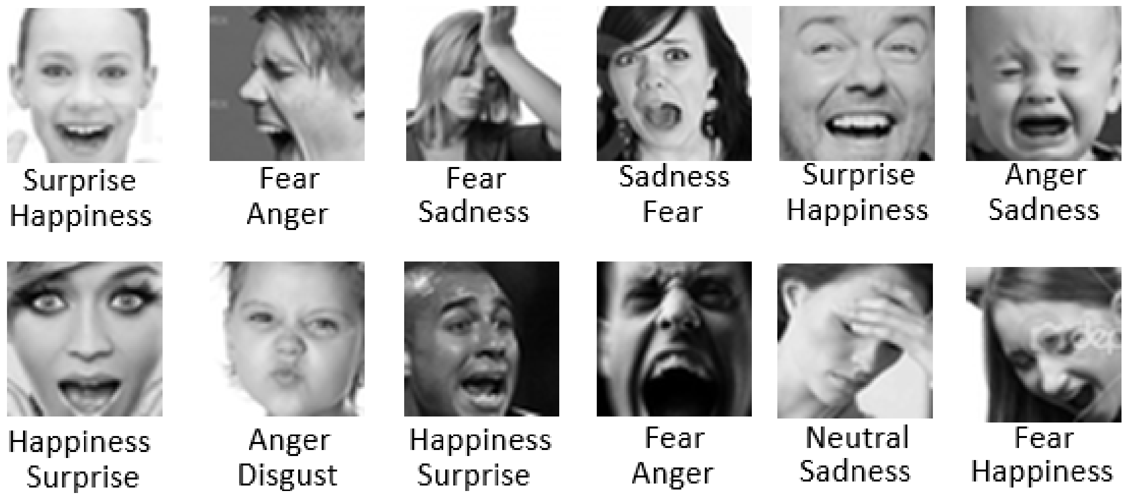


Figure 3.2: FER vs FER+ examples. Top labels are FER and bottom labels are FER+ (after majority voting).

the tagging quality.

In our first attempt, tagging was stopped as long as two taggers agreed upon a single emotion. But the obtained quality was unsatisfactory. So in the end, we asked 10 taggers to label each image, thus obtaining a distribution of emotions for each face image.

Figure 3.3 shows a plot relating the tagging quality versus the number of taggers. We randomly chose 10k images in the data set and assumed that the majority of the 10 labels were a good approximation to the "ground truth" label. Then, for subsets of fewer taggers, we computed how many of the majority still agreed with the "ground truth" emotion.

It can be seen from the figure that when there were 3 taggers, the agreement was merely 46%. With 5 taggers, the accuracy improved to about 67% and, with 7 taggers, the agreement improved to above 80%. With this, we concluded that the number of taggers has a high impact on the final label quality Rosenthal 2005.

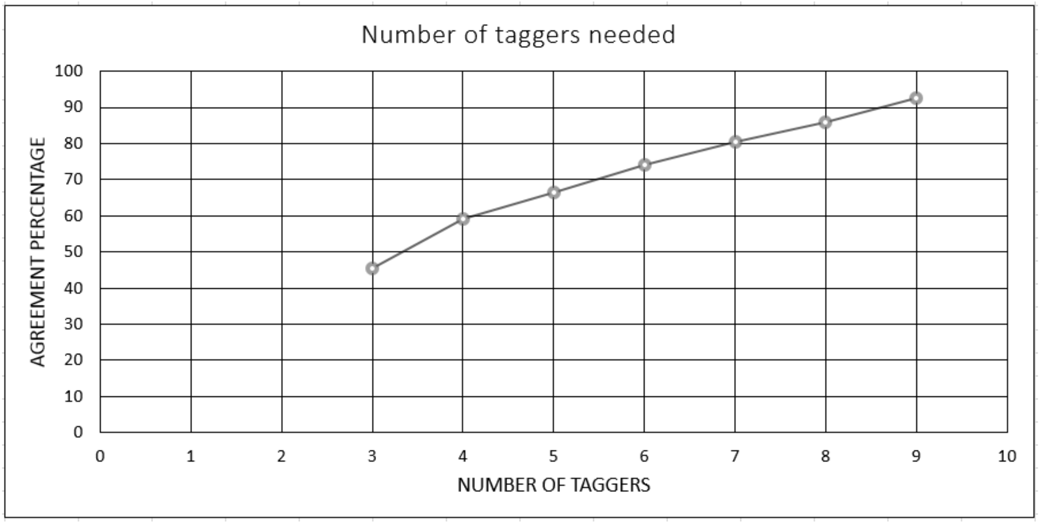


Figure 3.3: Tagger count versus quality.

With 10 annotators for each face image, we generated a probability distribution of emotion capture by the facial expression, which enabled us to experiment with multiple schemes during training. After first describing our machine learning architecture, we discuss in depth the four schemes that we tried for improving ground truth: majority voting, multi-label learning, probabilistic label drawing, and cross-entropy loss.

3.2.2 Network architecture

Discriminating emotion based on appearance is essentially an image classification problem. Therefore, a state-of-the-art Deep Convolutional Neural Network (DCNN) model that performs well in image classification should also perform well in facial expression recognition. We tried multiple DCNN models, including custom versions of the VGG network Simonyan and Zisserman 2014b, GoogLeNet Szegedy et al. 2015, ResNet He et al. 2015, etc. But since comparing different DCNN models was not the objective of our work, we adopted a custom VGG network to demonstrate emotion

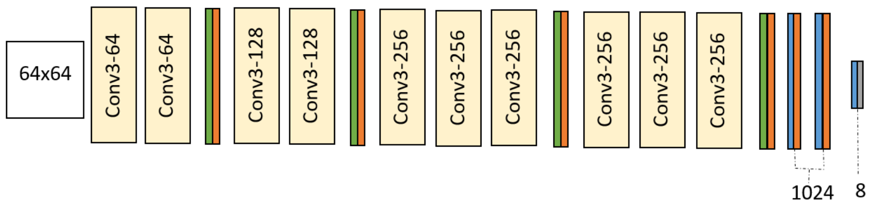


Figure 3.4: Our custom VGG13 network: yellow, green, orange, blue and gray are convolution, max pooling, dropout, fully connected and soft-max layer, respectively.

recognition performance on the FER+ data set.

The input to our emotion recognition model was a gray scale image at 64×64 resolution. The output was 8 emotion classes: neutral, happiness, surprise, sadness, anger, disgust, fear and contempt. Our custom VGG13 model is shown in Figure 3.4. It had 10 convolution layers, interleaved with max pooling and dropout layers. More specifically, after the input layer, there were 2 convolution layers with 64 kernels of size 3×3. After max pooling, a dropout layer was added with a dropout rate of 25%. The structure repeated but changed in the number of convolution layers and number of kernels. After all the convolution layers, 2 dense layers were added, each with 1024 hidden nodes, followed by a 50% dropout layer. The final dense layer was followed with a soft-max layer to generate the output.

Although the FER+ training set had only about 35k images, the dropout layers were effective in avoiding model overfitting in our model.

3.2.3 Training

We trained the custom VGG13 network from scratch on the FER+ data set employing the same split between training, validation, and testing data as provided in the original FER. During training we augmented the data set on the fly, applying affine transforms similar to those in Yu and Zhang 2015. Such data augmentation has been proven to improve the robustness of the model against translation, rotation and scaling.

Thanks to the large number of taggers per image, we could generate a probability distribution for each face image. We examined how to utilize the label distribution in a DCNN learning framework during training.

Let there be a total of N training examples $\mathbf{I}_i, i = 1, \cdots, N$. For the i^{th} example, let the custom VGG13 network's output after its soft-max layer be q_k^i, $k = 1, \cdots, 8$, and the crowd-sourced label distribution for this example be p_k^i, $k = 1, \cdots, 8$. Naturally, we have:

$$\sum_{k=1}^{8} q_k^i = 1; \quad \sum_{k=1}^{8} p_k^i = 1. \tag{3.1}$$

We experimented with four different schemes: majority voting (MV), multi-label learning (ML), probabilistic label drawing (PLD) and cross-entropy loss (CEL).

3.2.3.1 Majority voting

In most existing facial expression date sets, each facial image is only associated with one single emotion. It is natural to use the majority of the label distribution as the single tag for the image. More formally, we created a new target distribution $\hat{p}_k^i$ for each example $\mathbf{I}_i$, such that:

$$\hat{p}_k^i = \begin{cases} 1 & \text{if } k = \arg\max_j p_j^i \\ 0 & \text{otherwise} \end{cases}. \tag{3.2}$$

The cost function for DCNN learning is the standard cross-entropy cost, i.e.

$$\mathcal{L} = -\sum_{i=1}^{N}\sum_{k=1}^{8} \hat{p}_k^i \log q_k^i. \tag{3.3}$$

3.2.3.2 Multi-label learning

Many face images may exhibit multiple emotions. For example, someone can be happily surprised, or angrily disgusted. The idea of multi-label learning is to admit that such multi-emotion cases exist, and it was fine for our learning algorithm to match with any of the emotions that had sufficient number of taggers labeling them. Mathematically, we adopted a new loss function as follows:

$$\mathcal{L} = -\sum_{i=1}^{N} \arg\max_k I_\theta(p_k^i) \log q_k^i, \tag{3.4}$$

where $I_\theta(p_k^i)$ is an indicator function with threshold θ:

$$I_\theta(p_k^i) = \begin{cases} 1 & \text{if } p_k^i > \theta \\ 0 & \text{otherwise} \end{cases}. \tag{3.5}$$

Since more than one emotion is acceptable for each face, we let the algorithm pick the emotion it wants to train on, based on the output probability of each emotion. This was basically applying multi-instance learning in the label space. Effectively, as long as the network output agreed with any of the emotions that a certain portion of the taggers agree, the cost would be low. In our experiments, the parameter θ is set to 30%.

Table 3.1: Testing accuracy from training VGG13 using four different schemes: majority voting (MV), multi-label learning (ML), probabilistic label drawing (PLD) and cross-entropy loss (CEL).

Scheme	Trials					Accuracy
	1	2	3	4	5	
MV	83.60 %	84.89 %	83.15 %	83.39 %	84.23 %	83.852 ± 0.631 %
ML	83.69 %	83.63 %	83.81 %	84.62 %	84.08 %	83.966 ± 0.362 %
PLD	85.43 %	84.65 %	85.34 %	85.01 %	84.50 %	**84.986 ± 0.366** %
CEL	85.01 %	84.59 %	84.32 %	84.80 %	84.86 %	84.716 ± 0.239 %

3.2.3.3 Probabilistic label drawing

In the probabilistic label drawing approach, when an example is used in a training epoch, a random emotion tag is drawn from the example's label distribution p_k^i. We then treated this example as if it had a single emotion label as the drawn emotion tag. In the next epoch, the random drawing would happen again, and may be associated with a different emotion tag. Over the multiple epochs during training, we believed we would approach the true label distribution on average. Formally, at epoch t, we created a new distribution $\tilde{p}_k^i(t)$:

$$\tilde{p}_k^i(t) = \begin{cases} 1 & \text{if } k = \text{choice}(p_j^i) \\ 0 & \text{otherwise} \end{cases}, \tag{3.6}$$

where $\text{choice}(p_j^i)$ is a random number generator based on the distribution p_j^i. The cost function for DCNN is the same standard cross-entropy loss:

$$\mathcal{L}(t) = -\sum_{i=1}^{N} \arg\max_k \tilde{p}_k^i(t) \log q_k^i. \tag{3.7}$$

3.2.3.4 Cross-entropy loss

The fourth approach was the standard cross-entropy loss. We treated the label distribution as the target we want the DCNN to approach. That is:

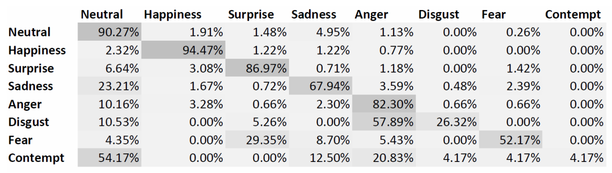
$$\mathcal{L} = -\sum_{i=1}^{N}\sum_{k=1}^{8} p_k^i \log q_k^i. \tag{3.8}$$

3.2.4 Experimental results

We tested the above four schemes on the FER+ data set we created. As mentioned earlier, each image was tagged by 10 taggers. The label distribution was generated with a simple outlier rejection mechanism: if an emotion was tagged less than once, the frequency count of that emotion was reset to zero. The label frequencies were normalized to ensure the distribution sum to one.

To compare the performance across all four approaches on the test set, we took the majority emotion as the single emotion label, and we measured prediction accuracy against the majority emotion.

For each scheme, we trained our custom VGG13 network 5 times, and reported the accuracy numbers, as in Table 3.1. Due to random initialization, the accuracy of the same scheme varied across different runs. It can be seen from the table that the PLD and CEL approaches yielded the best accuracy on the test data set. Both approaches were over 1% better in accuracy compared with MV. The t-value was around 3.1, which gave a probability of 99%-99.5% that the statistic was significant. On the other hand, the difference between PLD and CEL was within the standard deviation.



	Neutral	Happiness	Surprise	Sadness	Anger	Disgust	Fear	Contempt
Neutral	90.27%	1.91%	1.48%	4.95%	1.13%	0.00%	0.26%	0.00%
Happiness	2.32%	94.47%	1.22%	1.22%	0.77%	0.00%	0.00%	0.00%
Surprise	6.64%	3.08%	86.97%	0.71%	1.18%	0.00%	1.42%	0.00%
Sadness	23.21%	1.67%	0.72%	67.94%	3.59%	0.48%	2.39%	0.00%
Anger	10.16%	3.28%	0.66%	2.30%	82.30%	0.66%	0.66%	0.00%
Disgust	10.53%	0.00%	5.26%	0.00%	57.89%	26.32%	0.00%	0.00%
Fear	4.35%	0.00%	29.35%	8.70%	5.43%	0.00%	52.17%	0.00%
Contempt	54.17%	0.00%	0.00%	12.50%	20.83%	4.17%	4.17%	4.17%

Figure 3.5: Confusion matrix for the probability scheme.

The slight advantage of PLD was explained by its similarity to the independently discovered DisturbLabel approach in Xie et al. 2016.

It was a bit surprising to us that ML did not achieve as good performance as PLD and CEL. Since we asked each tagger to tag only the single dominate emotion, we expected that the label distribution did not necessarily reflect the emotion distribution of the underlying image. We also thought ML would be a more flexible learning target. We hypothesized that it might be because during testing only the majority emotion was used, and there was a bigger mismatch between training and testing for ML.

Figure 3.5 shows the confusion matrix of the best performing network. We performed well on most of the emotions except disgust and contempt. This was because we had very few examples in the FER+ training set that were labeled with these two emotions.

3.2.5 Summary

In this section, we compared different schemes of training a DCNN on crowd-sourced label distributions. We showed that taking advantage of the multiple labels per image boosted the classification accuracy compared with the traditional approach of a single label from majority voting. We have made the FER+ data set available for download[1].

These same techniques can be beneficial for video understanding in general, and not just emotion recognition. This is because the problems in video classification are two-fold:

1. Video clips duration usually vary a lot, from several seconds to several minutes long. One class label per clip is not enough, a lot of small segments are common to more than one class.
2. Some of the existing web datasets are only weakly labeled, which negatively affects classification performance.

We choose emotion recognition to experiment with different labeling and training strategies, instead of video, because of cost, including the cost of crowd sourcing and the emotion recognition project was funded. Nevertheless, having multiple labels per clip or per segment in action recognition enables us to train better classifiers.

3.3 Feature extraction

The first step in video recognition is to extract per-frame features. These are then fed into another pipeline, in order both to make sense of them, and to take the time context into consideration. Depending on the amount and type of data, feature extraction is usually done using a CNN model that has already been trained on ImageNet or another large dataset. For depth data, sometimes we still use hand-crafted features, due to their limited size and the lack of a good pre-trained depth

[1]`https://github.com/Microsoft/FERPlus`

model. However, can also use RGB pre-trained models on depth data, with minor changes, if depth is interpreted as a gray scale intensity.

3.3.1 Color features

Many approaches have been tried for color. They fall into the following groups.

3.3.1.1 Simple pre-training

Most publications use the last Fully Connected (FC) layer from a pre-trained CNN model, usually called FC7 or FC6 before the softmax layer, as the per-frame feature. Such a layer is usually a 2048- or 4096-length float vector, depending on the network. However, the problem in this approach is that it loses input spatial information.

3.3.1.2 Region-based

Another approach that has been used recently is to use the last convolution layer vector as the feature, as before, but then pass the result to a visual attention mechanism Sharma, Kiros, and Salakhutdinov 2015. The convolution layer output is used to create a spatially organized feature map, one that splits the input image into a grid, and provides a feature per cell inside the grid. As shown in Figure 3.6, the output of the convolution layer now populate a 3D tensor instead of a vector, and the weighted value of that cube is fed to a two-layer LSTM. The weighting of the LSTM represents a form of soft attention that is learned during training, providing coarse spatial information. A related approach combines the output of multiple layers and fuses them together to form feature vectors that cover coarse, medium and fine details.

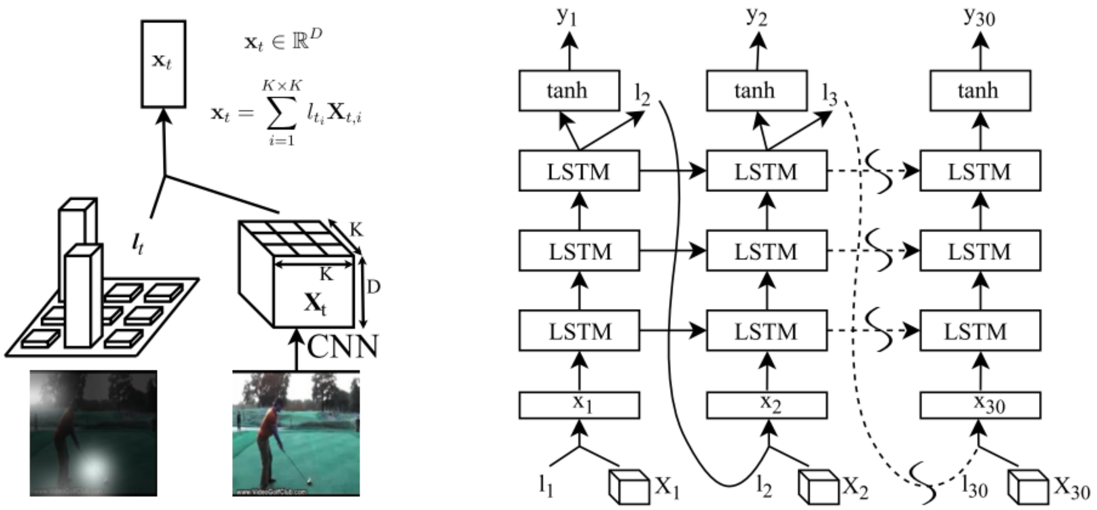


Figure 3.6: From Sharma, Kiros, and Salakhutdinov 2015, Visual Attention for action recognition.

3.3.1.3 Object-based

Another type of feature that is also CNN based and used primarily in caption generation is a pre-trained Region-Based Convolutional Neural Network (R-CNN) Ren et al. 2015 and its variants. R-CNN takes a single image and generates a number of regions from the image with their classification and location. The advantage of this approach is that the region corresponds to known objects, with their feature vectors and their locations in the image. The main disadvantage is that R-CNN does not capture the actual action.

3.3.1.4 Action-based

Therefore, an important feature for action recognition is 3DCNN Ji et al. 2013. The idea here is to extend CNN by adding time as an additional axis. This approach allows us to learn the local dynamic between frames. The main problem of 3DCNN is that it cannot be trained on still images, so we cannot take advantage of the currently available image data. However, we now have large labeled video datasets such as Sport1M Karpathy et al. 2014 and YouTube 8M *Youtube 8M* 2016 (accessed December 17, 2016), which we can train 3DCNN on, given enough resources. The main drawback of those large video datasets is that they are currently weakly labeled. That is, they use whatever tag was associated with the corresponding video that was set by the uploader. These tags are usually not accurate, and the same action can be described by several different tags.

3.3.2 Depth features

Since the introduction of the Kinect, much action recognition research has been done on depth sensing, using skeleton data, or depth data directly, or a combination of both.

One big advantage of depth data is that they are not affected by the ambient light. Therefore, the quality of the data is the same with or without light (with some exception), which is a huge benefit. But because of the lack of large labeled datasets compared to its RGB count part, many publications use hand-crafted features tailored to depth data. The three most used techniques to extract features from depth data follow.

3.3.2.1 Manual

These use hand-crafted features tailored to depth data, mostly by extending existing RGB techniques such as SIFT, HOG, SURF, GLOH, and many others. Since with depth frames it is easier to separate the foreground from the background, in many applications we can then work on the motion blob only, and not the entire frame.

3.3.2.2 Pre-trained

These use a pre-trained model similar to RGB video. We convert depth data into a gray level image and triplicate it in the three color channels. This gray scale image is fed to a model pre-trained on ImageNet data. Surprisingly, this techniques works well in practice, most probably because these deep learning systems encodes spatial shape information independently of color.

3.3.2.3 3D cloud

These create a 3D cloud from the depth data, and then use a CNN for feature extraction, as shown in Figure 3.7. However, using 3D cloud data directly is impractical, due to the size of the data, and the size of the network needed to process them. Haque, Alahi, and Fei-Fei 2016 instead uses glimpses to focus on patches of the data, instead of the entire data at once.

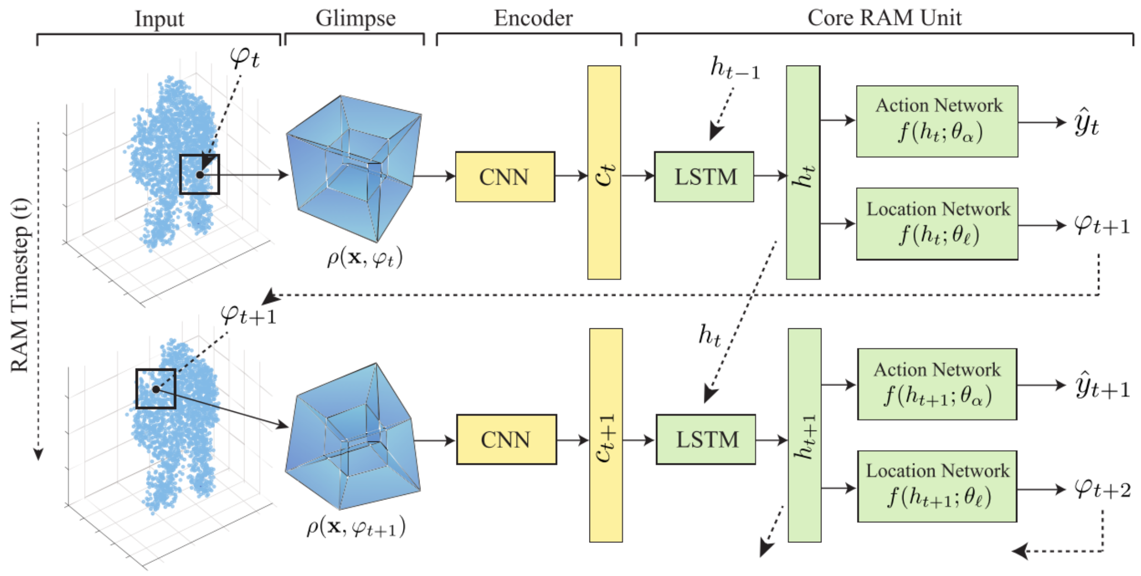

Input
Glimpse
Encoder
Core RAM Unit
RAM Timestep (t)
φ_t
$\rho(\mathbf{x}, \varphi_t)$
CNN
c_t
h_{t-1}
LSTM
h_t
Action Network $f(h_t; \theta_\alpha)$
$\hat{y}_t$
Location Network $f(h_t; \theta_\ell)$
φ_{t+1}
φ_{t+1}
$\rho(\mathbf{x}, \varphi_{t+1})$
CNN
c_{t+1}
h_t
LSTM
h_{t+1}
Action Network $f(h_{t+1}; \theta_\alpha)$
$\hat{y}_{t+1}$
Location Network $f(h_{t+1}; \theta_\ell)$
φ_{t+2}

Figure 3.7: From Haque, Alahi, and Fei-Fei 2016, Glimpses from 3D Cloud.

3.3.3 Skeleton features

For derived skeleton features, most publications use a normalized or pre-processed version of the actual skeleton joints directly as a feature vector. The pre-processing is usually a trivial step that simply maps joint coordinates into a camera independent coordinate system. For better training, it is usually recommended to map the range of each joint to $[-1, 1[$ or $[0, 1[$. This can be achieved by one of two techniques.

3.3.3.1 Camera-based

From the physical camera properties, we know the frustum dimensions and the possible ranges of each coordinate. With this information we can easily normalize the input data to the range needed by the model. However, the data usually does not use the entire recording area.

3.3.3.2 Data-based

By computing the means and standard deviations, or the minmaxes, of the entire dataset, we can use those values to normalize the input data. This gives a tighter fit, and avoids any dependency on camera properties. This was our preferred method.

3.3.4 Summary

In this section, we detailed many of the different techniques for feature extraction from RGB-D data. Feature extraction is crucial for model convergence, especially when dealing with a challenging network such as GAN. Furthermore, the network design depends on the method used to extract feature.

Based on multiple experiments, in our work we used a 2D pre-trained ImageNet model for both RGB and depth data, In order to take advantage of the pre-training models with depth data, we converted each depth frame to a gray level image with range $[0, 255]$, then stacked the frame three times, one for each channel. And, again based on multiple experiments, in our work we used statistics for skeleton data normalization. Particularly in our Human Prediction via Generative Adversarial Network (HP-GAN), we found that we needed to be very careful in the normalization of each skeleton, or else the networks diverged.

3.4 Video classification techniques

Still imagery is generally not enough to capture or describe action, except in some cases where it represents the peak of the action, and the peak snapshot is enough to describe the entire action. Having temporal information is crucial in order to recognize the action being performed and be able to localize it. Most actions can be described in three phases: start, which is the transition to the action, or the preparation for the action, from a neutral pose; peak, the distinguishing part of the action; end, the transition to a steady state, or the finishing of the action.

In this section, we describe the different approaches for taking temporal structure into consideration in the context of video recognition.

3.4.1 Simple statistics

In these methods, we treat the per-frame features as a simple bag of features, and combine them using some statistics in order to create the final descriptor of the given sequence. The final descriptor is then fed into a classifier, mostly a SVM, to classify the action. Statistics can be as simple as max or average pooling, or a combination of mean, average, variance, and higher order moments, or other more advanced techniques such as Fisher vectors and their variants.

Although using statistical information ignores sequence order and time information, it has been shown that, in many video analysis competition, statistical metrics provided a very competitive result. We showed that in our own submission Bargal et al. 2016 for the video based emotion recognition competition (EmotiW): using statistical metrics beat a RNN when the number of video clip was small. The reason is that a RNN-based model requires much data to train properly and generalize. As shown in Figure 3.8, we combined multiple pre-trained models together and used a

SVM for classification. We had tried different RNN-based models and had combined them with statistics, however, none were as successful, due to the lack of training. Our entry ranked third in the competition.

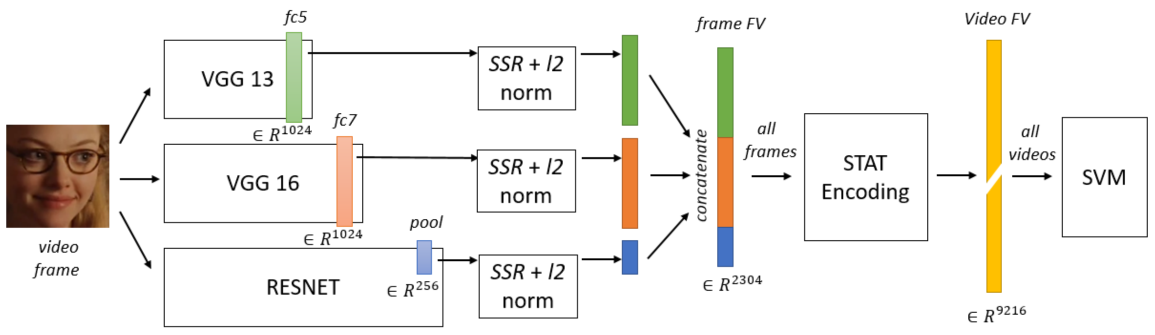


Figure 3.8: From Bargal et al. 2016, statistical based pipeline for emotion from video.

Using statistics does not mean ignoring the sequence order; you can implement statistical methods that incorporate it. Fernando and Gould 2016 developed a pooling technique that preserves the order of frames in the sequence, which they call "rank pooling". Although this approach is better than standard statistical approaches, it does not try to understand the temporal dynamics of the action.

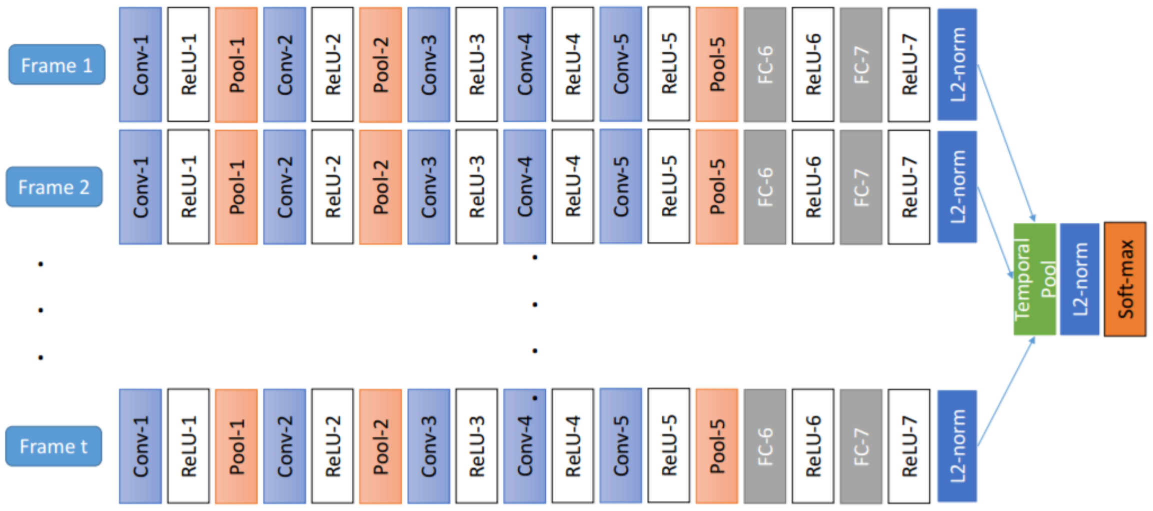


Figure 3.9: From Fernando and Gould 2016, temporal pooling.

3.4.2 Recurrent Neural Networks

RNN is a type of neural network that takes a sequence and outputs a sequence, while maintaining an internal state. Because of such behavior, RNNs have been used to learn the temporal dynamics of an input signal and have been applied successfully on many time based problems.

The basic RNN equations are as follow:

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$

$$o_t = \sigma_o(W_o h_t + b_o)$$

Where x_t is the input to the network at time t, h_t is the hidden state at time t, o_t the output at time t, f_h and f_o are non-linear functions for the hidden and output, and W_h, U_h and W_o are the corresponding trainable weights of the network.

The above network cannot learn long sequences due to the problem of vanishing and exploding gradients. Most publications use a LSTM, a Gated Recurrent Unit (GRU), or one of their variants. Those variations tackle the problem of vanishing gradients by providing a path for the gradient to back-propagate through time.

Here the equations of a LSTM:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$

$$h_t = o_t \circ \sigma_h(c_t)$$

The main idea from the above equations is that we learn to forget, using f_t forget gate, or add information, using i_t input gate, for the internal cell c_t. The internal cell c_t is the storage unit of LSTM.

For action recognition, a sequence of frame features is fed to a RNN and the sequence outputs are fused together somehow for the final classification. There are multiple ways to fuse the outputs.

1. Final frame: This method uses the last output of the sequence as the one for prediction. This last output only happens after the RNN has processed the entire sequence.
2. Mean pool: This method averages all the outputs in the sequence, with equal weighting over time.
3. : Temporal attention: This method weights the average of all the outputs using a soft attention mechanism. The attention weights are learned during training.

We have implemented and tested these three approaches, and found that attention weighting provided the best result by at least 2%. The reasons are that using the last temporal output will bias the result toward the last couple of frames. And, using the average will treat all frames equally, which often does not make sense for many actions; furthermore, the internal state of the network during the first couple of frames has not had time to learn anything yet.

We note that another approach that could potentially improve these results is the Bidirectional RNN, which combines two RNNs, one from start to end and another from end to start. Bidirectional RNNs can be applied with any of the above three methods. However, the main problem with Bidirectional RNNs is that they do not work in a streaming scenario, so we have not explored them.

3.4.3 Temporal attention models

Most attention models used in vision were inspired by Bahdanau, Cho, and Bengio 2014. The idea is simple: combine the output at each time step with different weights, in order to minimize the final loss. So the network needs to compute the importance of each input at time t during training in order to achieve the best possible performance.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$

$$h_t = o_t \circ \sigma_h(c_t)$$

$$\alpha_t = \frac{exp(o_t)}{\sum_{i=1}^{T} exp(o_i)}$$

$$atto = \sum_{t=1}^{T} \alpha_t h_t$$

From the above, α_t depends on normalized o_t across the current sequence. There are many variants to the above equation depending on the domain, but the main idea is the same, which is learning the importance of each frame in order to minimize the loss. As we stated previously, LSTM with attention has provided us with the best results compared to other methods.

3.4.3.1 RNN aggregation methods

In Mirsamadi, Barsoum, and Zhang 2017, we compared different pooling algorithms on audio data, in order to classify the emotion associated with each audio clip. In figure 3.10, the six methods are shown as: (a) Temporal aggregation using statistics, (b) Per-frame training, assuming that each frame has the same emotion, (c) Use of last frame and first frame in the case of bidirection, (d) Averaging across time, (e) Use of attention, with logistic regression, (f) Use of attention, with a FC network, which replaces logistic regression with a more complex network.

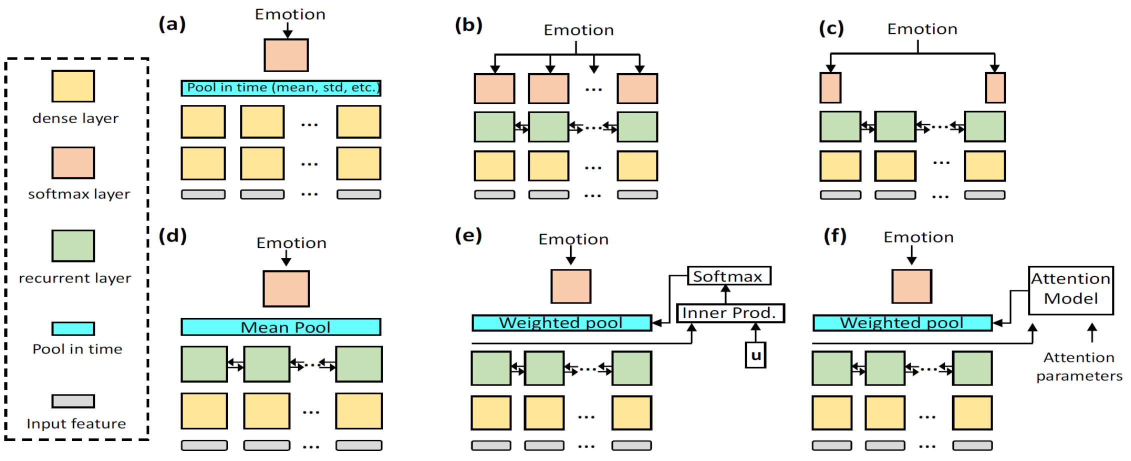


Figure 3.10: From Mirsamadi, Barsoum, and Zhang 2017, comparison of various pooling techniques for emotion recognition from audio.

Figure 3.11 shows the learned weights from the attention model. The network had learned where the gaps were in the audio clip, usually due to background noise or background audio; they are not easy to detect or identify. Also, peak weights corresponded to peaks in the emotion audio.

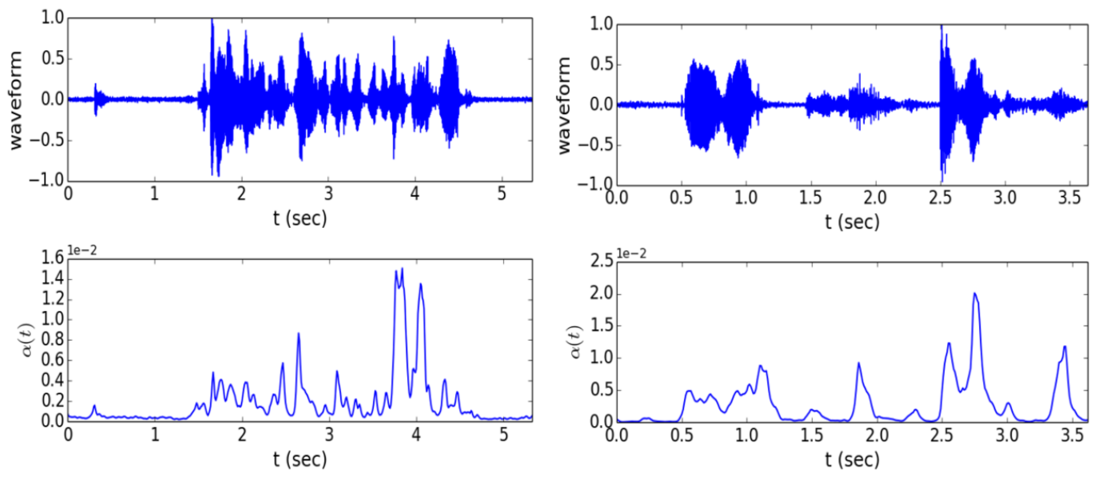


Figure 3.11: From Mirsamadi, Barsoum, and Zhang 2017, the result of emotion recognition using attention model.

In table 3.2, we show the accuracy under different temporal aggregation method.

Features	Temporal aggregation	Accuracy
Raw spectral	frame wise	57.7%
	final frame	54.4%
	mean pool	56.9%
	temporal attention	61.8%
Feature	frame wise	57.2%
	final frame	53.0%
	mean pool	62.7%
	temporal attention	63.5%

Table 3.2: From Mirsamadi, Barsoum, and Zhang 2017, the result of emotion recognition using different aggregation modes.

The frame-wise method, a baseline, simply assumed that each frame had the same emotion, and used softmax for each frame. The final-frame method only considered the last frame of the recurrent network, since at the final frame the network internal state had already seen all previous frames, and therefore had enough context to classify the clip. The mean-pool method simply added a temporal mean layer before the softmax. And finally, the temporal-attention method learned which utterance was more important than the others. As shown in the figure, independently of using raw data or hand-crafted features as input, the temporal-attention method produced the best accuracy.

3.4.4 Neural Networks with external memory

A Neural Network with external memory adds read-write memory access to the Neural Network, expanding the capacity of the network, and providing an added capability that it is difficult to achieve with a vanilla Neural Network. The network then primarily acts as the controller of a Neural Turing Machine (NTM). The network can be any type of neural network, but our focus is RNNs and their variants. The RNN understands vectors, so the external memory bank used is usually an array of vectors. There are two main approaches to learn where and how to access the memory during training:

1. Differential access: This uses the soft attention mechanism, so it can read-write in all locations at once with different weights at each location. The advantage of this approach is that it allows the use of back propagation.
2. Using Reinforcement Learning (RL): access one location at a time, in this case it is not differential so we can not use back-propagation as the previous option. The most used technique for non-differential training is reinforcement learning.

There are at least three benefits of a NTM over the more standard LSTM. First, it decouples computation from storage, so we can increase storage while keeping computation low. Second, its architecture is close to a standard computer architecture, but more easily learnable from data. Third, it can memorize a larger chunk of data in a sequence.

For video, one of the first uses of memory attention in video was Fakoor et al. 2016. They used attention with memory to generate descriptions from video data.

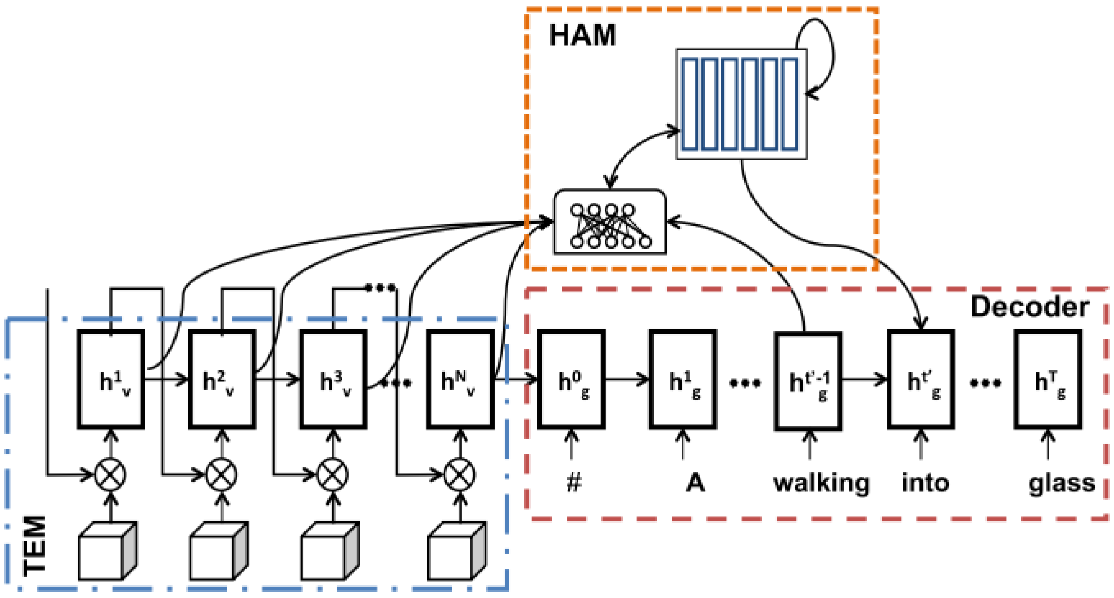


Figure 3.12: From Fakoor et al. 2016, Memory Attention for Video Description.

As shown in Figure 3.12, their Temporal Modeler (TEM) encodes the temporal structure of the video, using visual attention for each frame. Their Hierarchical Attention Memory (HAM) keeps track of all previous attentions.

3.4.5 Summary

In this section, we reviewed various existing approaches for video classification, from ignoring the temporal information to the various techniques that incorporate and learn temporal dynamics. For small dataset, statistics, pre-trained models and hand-crafting features are the best for video recognition. As the size of the data increase, more advance techniques can used to take advantage of the causal effect. In this thesis, we are tacking advantage of the time information and we are using temporal attention in our training to let the network learn which part of the stream is more important.

4 Prediction and unsupervised learning

In this chapter, we focus explicitly on unsupervised learning methods, since this thesis relies heavily on GANs.

Humans do not learn through supervised learning; they can generalize from very few examples without the need of detailed labeling of every object. Many researchers Lake et al. 2017 have been trying to mimic such basic human intelligence. The two most promising directions in closing the gap between how humans learn versus how machines learn *The Missing Link of Artificial Intelligence* are through the methods of unsupervised learning and reinforcement learning. Our work uses and extends the state of the art of the first.

Unsupervised learning using neural networks is a decades-old idea Carpenter and Grossberg 1988; Kohonen 1982, and there has been much research in unsupervised learning in and out of neural networks. However, recently a new type of unsupervised learning called GAN Goodfellow et al. 2014 is gaining research attention. The GAN concept was inspired by the MinMax algorithm from game theory. It opposes two networks: a generative network, which tries to generate new “fake” data as accurately as possible to real data, and a discriminator network, which tries to distinguish between fake and real data. Both networks get trained in parallel, and both try to outdo each other. We will explain why this approach is very useful for action recognition, and why we have adopted it.

4.1 Generative Adversarial Nets

With the proliferation of motion capture devices and multimedia social apps, we now have a vast amount of unlabeled multimedia contents that are impractical to label manually. But with the availability of high-end GPUs and custom chips for deep learning, the processing large amount of data became a tractable possibility. Unsupervised training enables us to gain knowledge from available raw data without the need of expensive and errorful manual labelingg. Transferring this unsupervised knowledge, in order to speed up supervised training, will help reduce the needed amount of labeled data.

The first paper on GANs was published in 2014 by Goodfellow et al. 2014 on still images. The main idea was to have two networks, a generative network that generates images, and a discriminator network that classifies the input as a real image or fake image. Assume that the discriminator network is $D(x)$ and the generative network is $G(z)$. $D(x)$ takes an image x and outputs the probability that the image is real or not. $G(z)$ takes a random vector z and generates an image. Below is the cost function for both networks.

$$L_D = logD(x) + log(1 - D(G(z))$$

$$L_G = log(1 - D(G(z)))$$

The training alternates between the above cost functions: L_D maximizes $D(x)$ and minimizes $D(G(z))$, whereas L_G maximizes $D(G(z))$

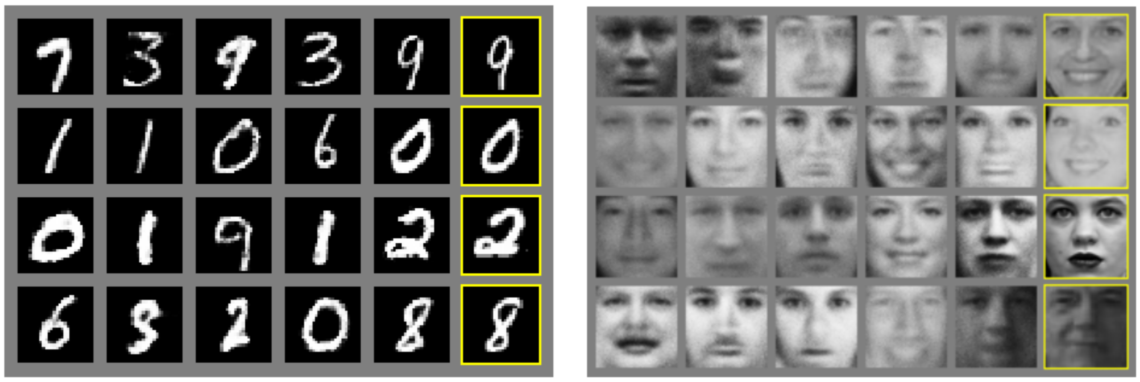


Figure 4.1: From Goodfellow et al. 2014, all columns except the right most one are generated images from GAN.

As shown in Figure 4.1, all the columns in the two sub-figures, except the right most column, are generated images from the generative net. Intuitively, this shows how powerful the generative network is.

One of the main advantages of a GAN, compared to other methods, is that the loss function of the generator is learned during training, and the discriminator network is already part of the generator loss function. Other methods for generator loss instead use L_2 loss or something similar, which does not generate realistic looking images due to the blurring effect of averaging.

Still more recent work Liang et al. 2017 combines the GAN approach with a VAE, in order to improve the prediction power of the model, which in GANs are based on z. Although we do not use a VAE directly in this thesis, we have taken inspiration from this combination in our construction our own modified GANs.

The VAE is a more refined approach for training the probabilistic generative model. VAEs extend Autoencoder (AE)s by constraining the output of the encoder to a Gaussian distribution, and then updating the decoder to sample values from this output probability. The encoder simply generates 2 vectors for each variable, according to the mean and variance of a high-dimensional Gaussian probability distribution. The decoder then samples a value from each variable, and uses this value to reconstruct the original input using L_2 distance.

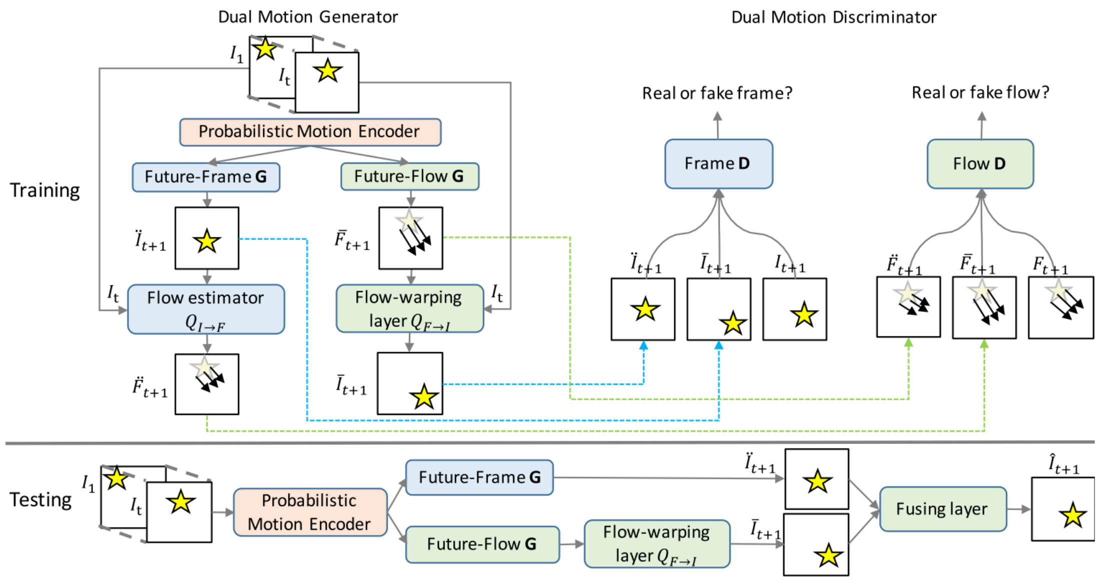


Figure 4.2: From Liang et al. 2017, hybrid VAE + GAN network.

4.2 Pixel-based prediction

We can use GAN-like algorithms on video in trying to predict future frames. And, in doing so, the network learns the temporal dynamics of the video, enabling us to use this learned information to improve video recognition. The idea is simple: the generative network will take a sequence of frames as input, and predict a sequence of future frames as output. The discriminator network will take a sequence of frames as input, and return the probability that the last couple of frames are future frames. If we succeed in predicting future frames, then the model is powerful enough to understand the internal dynamics of the video stream.

The above technique has been demonstrated in Mathieu, Couprie, and LeCun 2015. They used a Laplacian Pyramid representation in their generative and discriminator networks, in order to keep the predicted frame resolution similar to the input, as shown in Figure 4.3.

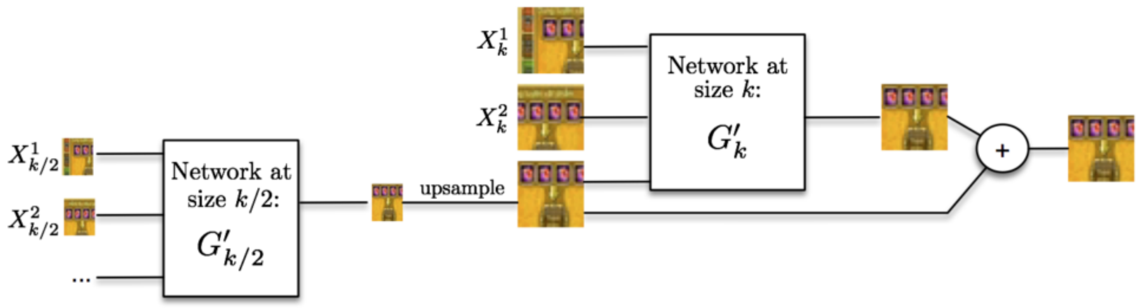

$X^1_{k/2}$
$X^2_{k/2}$
...
Network at size $k/2$: $G'_{k/2}$
upsample
X^1_k
X^2_k
Network at size k: G'_k
+

Figure 4.3: From Mathieu, Couprie, and LeCun 2015, video generator network using Laplacian Pyramid method in order to upsample the output of each layer.

As for the loss function, they used the following:

$$L_D = \sum_{k=1}^{N_{scale}} L_{bce}(D_k(X_k, Y_k), 1) + L_{bce}(D_k(X_k, G_k(X_k)), 0)$$

$$L_G = \sum_{k=1}^{N_{scale}} L_{bce}(D_k(X_k, G_k(X_k)), 1)$$

where L_{bce} is the binary cross-entropy loss. The idea is very similar to the still image approach. The loss of the discriminator tries to make the discriminator return 1 if the input is a real sequence, and 0 otherwise. Meanwhile, the loss of the generator tries to make the discriminator return 1 on generated sequence. The training algorithm iterates between both loss functions.

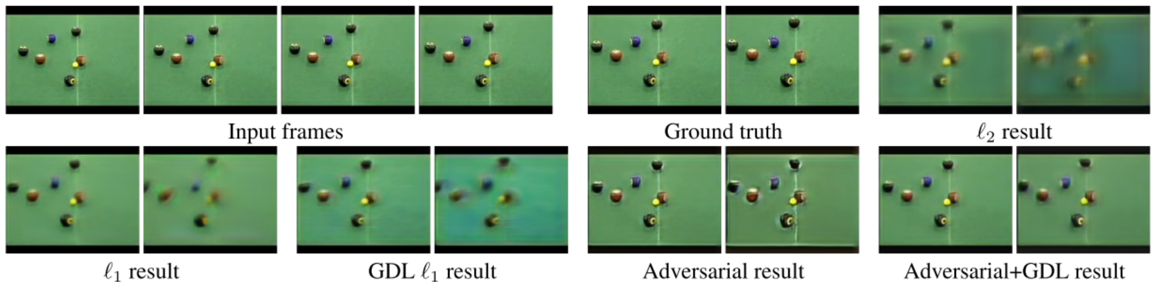


Figure 4.4: From Mathieu, Couprie, and LeCun 2015, video prediction comparison for a pool game. The purple ball moves a little from one frame to another.

Figure 4.4 compares the results of non-generative methods with those of a GAN for video frame prediction in a pool game. In this figure, the purple ball moves a little from one frame to another. As shown in the figure, the prediction from GAN using a Gradient Difference Loss (GDL) outperformed all other methods.

4.3 Object-based prediction

But predicting future frames at the pixel level is very difficult. This is why current state of the art results are blurry, and why they can only predict a few frames into the future before collapsing. For this reason, most research focuses on representations spaces that capture object motion properties, such as affine transformation of entire objects in two dimensions, or constrained 3D transformations of skeleton data.

In Liu, Barsoum, and Owens 2018, our research predicted full object motions within the Moving MNIST dataset Srivastava, Mansimov, and Salakhutdinov 2015. We did this by extending the dynamic routing algorithm of CapsNet Sabour, Frosst, and Hinton 2017 to detect the coordinates of the object in the image. CapsNet tries to preserve the spatial information that ordinarily gets lost in a standard convolution network with pooling. It does so by introducing the concept of a "capsule" vector for each object. Each element of the vector maps to certain properties related to the object it detects, and the magnitude of the vector is normalized to a range of $[0, 1]$ to represent the extent to which the object actually exists.

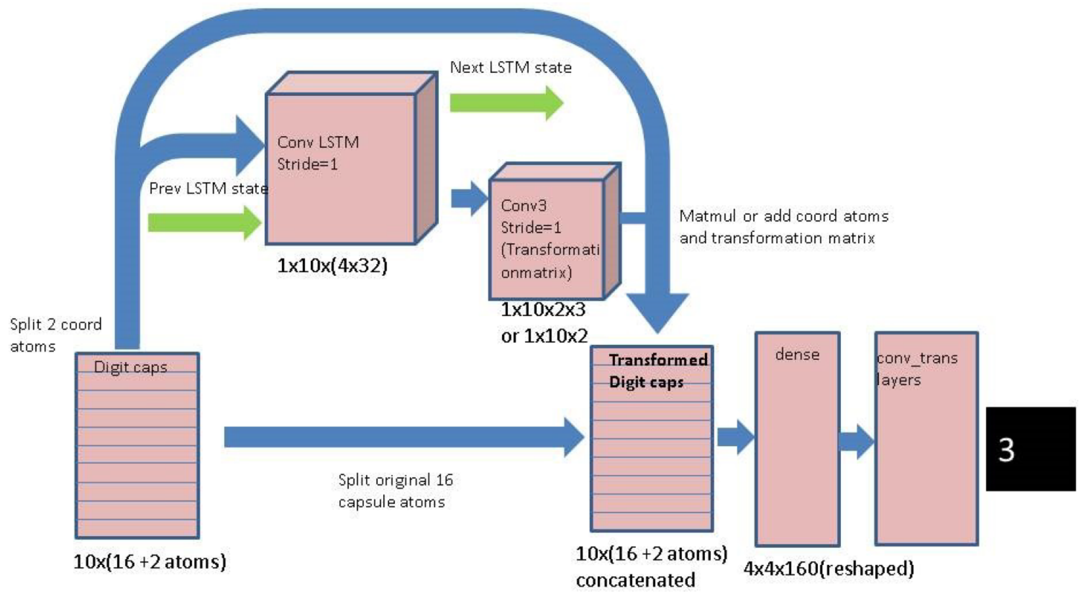
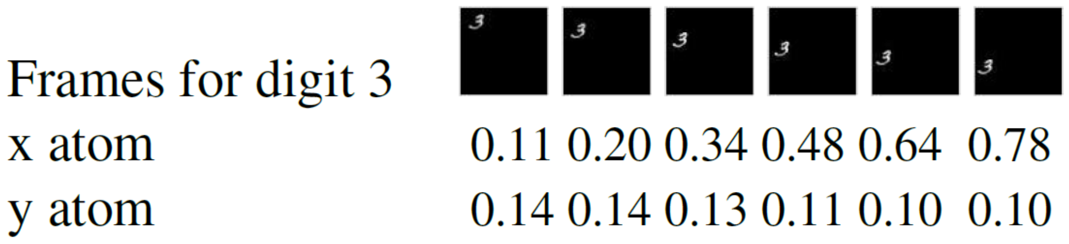


Figure 4.5: From Liu, Barsoum, and Owens 2018, predicting future position using an improved CapsNet.

We extended the CapsNet vector with two more elements, which learned the (x, y)

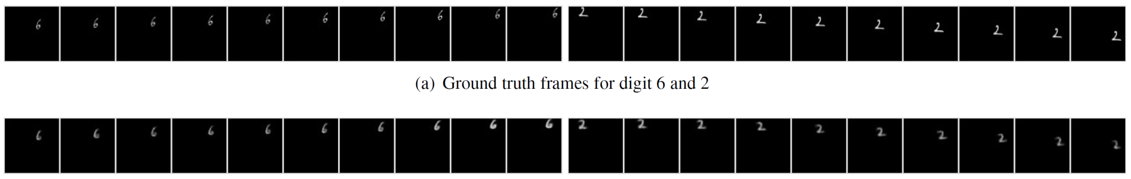
coordinates of the object being represented by a specific capsule. We then trained this new CapsNet to estimate the *present* location of the object in the image, as shown in figure 4.6.



Frames for digit 3						
x atom	0.11	0.20	0.34	0.48	0.64	0.78
y atom	0.14	0.14	0.13	0.11	0.10	0.10

Figure 4.6: From Liu, Barsoum, and Owens 2018, showing the detection of (x, y) coordinate for the number 3.

We then added another Convolutional LSTM network to predict the *future* location of the object in the image, as shown in figure 4.5. Both networks were trained separately. The result of the prediction network is shown in figure 4.7.

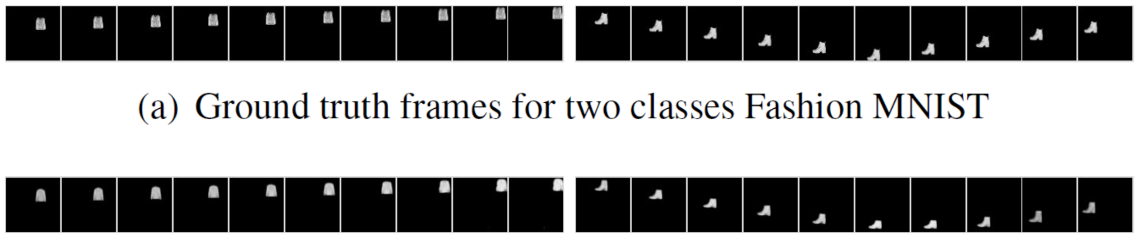


(a) Ground truth frames for digit 6 and 2

(b) Generated frames for digits 6 and 2 with affine transformations

Figure 4.7: From Liu, Barsoum, and Owens 2018, showing prediction of (x, y) coordinates for the numbers 6 and 2.

The primarily purpose of the CapsNet is to convert the entire object in the image into a point mass, as in the theories of kinematics, in order focus on its motion instead of its identity. To test this, we trained only the CapsNet network on another dataset, the Fashion MNIST dataset Xiao, Rasul, and Vollgraf 2017, and re-used the same predictive LSTM network that was trained on the original Moving MNIST dataset, without any fine tuning. The result of the prediction is shown in figure 4.8. As shown, even without retraining the predictor network on the new dataset, it still performed well.


(a) Ground truth frames for two classes Fashion MNIST

(b) Generated images for two classes Fashion MNIST

Figure 4.8: From Liu, Barsoum, and Owens 2018, showing transfer learniing the predictor of $(x, y($ coordinate for fashion MNIST datatset.

4.4 Unsupervised action recognition

If we can predict future human motions, then we learn the dynamics that govern human motion. With a trained generator that can generate future frames, and a discriminator that can discriminate between real versus fake sequence of frames, we can improve human action recognition in two ways:

- Discriminator re-use: The discriminator learns the difference between real and fake sequences of human motion. Therefore, we can use the features learned by the discriminator for other related tasks such as action recognition.
- Data augmentation: The generator learns to generate possible futures. Therefore, we can use the generator to generate more human poses, especially for classes in which we have little data, improving classifiers and reducing biases.

One major advantage of this approach is that we can accelerate learning. We can first train a GAN model on a large unlabeled dataset, and then fine-tune the resulting discriminator on a smaller labeled dataset.

4.5 Summary

Unsupervised training is getting a lot of attention lately, due to the current success of GAN and VAE. In this section we summarized current trends and some of our work and improvement we did in this area. Our primarily focus in this thesis is human motion prediction using GAN, the reason for selecting GAN instead of VAE is that in GAN there is no restriction of the probability distribution that we are trying to learn, and the actual loss is also learnable. As for VAE, we learn the parameters of a multivariate Gaussian distribution, therefore, we put constrains on the probability distribution that we are trying to learn.

5 Datasets

There are a number of large scale datasets for action recognition, especially for RGB video, such as UCF, Sport1M Karpathy et al. 2014, YouTube8M *Youtube 8M* 2016 (accessed December 17, 2016) from Google, and more recently SLAC Zhao et al. 2017(Sparsely Labeled ACtions) from MIT and Facebook. Some of those datasets are part of a yearly action recognition and detection competition, which enable researchers to compare new ideas with current state of the art results. Creating a large enough dataset for human motion recognition is costly and difficult, due to the amount of data that needs to be labeled, the difficulty of labeling those data consistenly, and the computation needed to process them. This is why most large scale video dataset have been created by large corporations collaborating with academia.

But since the introduction of the Kinect for Xbox, there has been an increasing interest in action recognition and human motion understanding from the RGB-D signal. Due to the richness of the amount of information in the depth stream, and their invariance to ambient light, the number of collected Kinect datasets has been increasing year after year. The current largest RGB-D dataset is called NTU

RGB+D Shahroudy et al. 2016b. It was released in 2016, and it is one of the first to use Kinect v2, instead of the Xbox 360 Kinect sensor; the quality of the capture is therefore higher than the original Kinect datasets.

Another important RGB-D dataset is Human3.6MIonescu, Li, and Sminchisescu 2011; Ionescu et al. 2014, which is currently one of the largest RGB-D datasets from motion capture (MoCap) sources. These MoCap datasets are usually more accurate and have less outliers compared to Kinect data.

Our research focus on the video understanding and anticipation of skeleton data. As such, we primarily focus on the NTU RGB-D and the Human3.6M data sources. Both sources contain RGB, Depth, IR, and skeleton clips for each action. With the very large sizes of both datasets, it gives us opportunity to experiment with multiple algorithms.

5.1 NTU RGB-D

NTU RGB-D contains 56880 clips (RGB-D, IR, RGB, and skeleton) and 60 classes from 40 subjects. Each action has been recorded from 3 different views. All data were captured using Microsoft Kinect v2. Each clip is from one to five seconds, so they tend to be rather short in length.

Below are some samples from this dataset. Some of the activities have one person only, but other activities have two people interacting together, such as Figure 5.3. We show only one color frame and the corresponding segmented depth frame; however, the actual dataset has color video, a full depth clip, IR video, skeleton joints, and segmented depth.

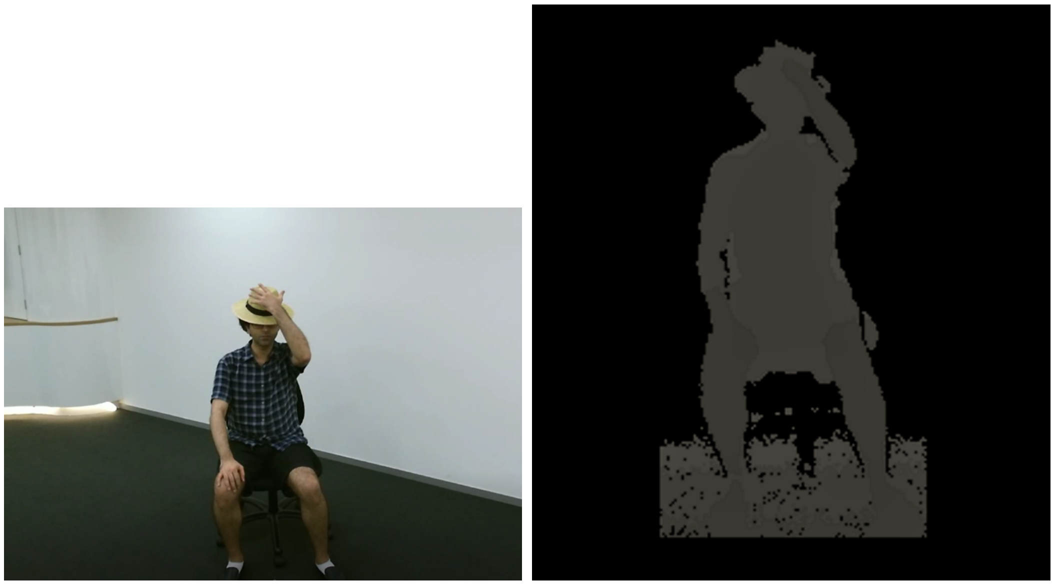


Figure 5.1: Putting on hat.

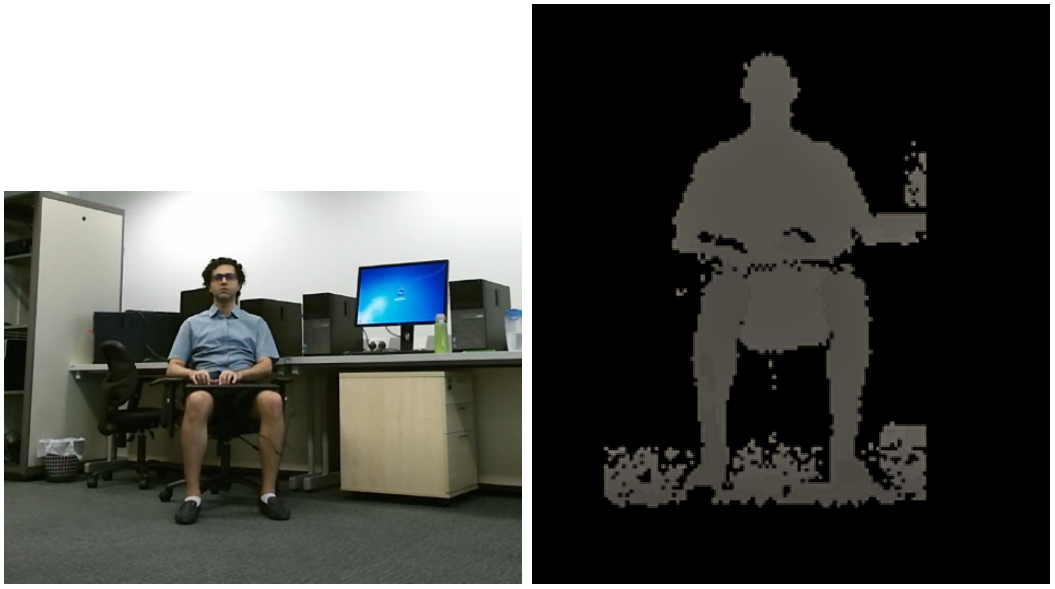


Figure 5.2: Typing on the keyboard.

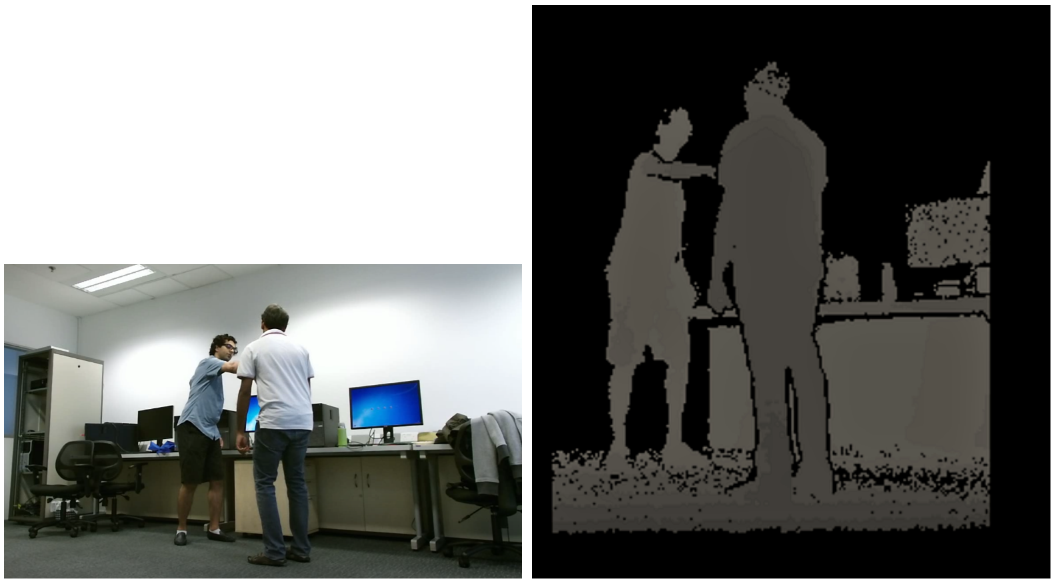


Figure 5.3: Punching another person.

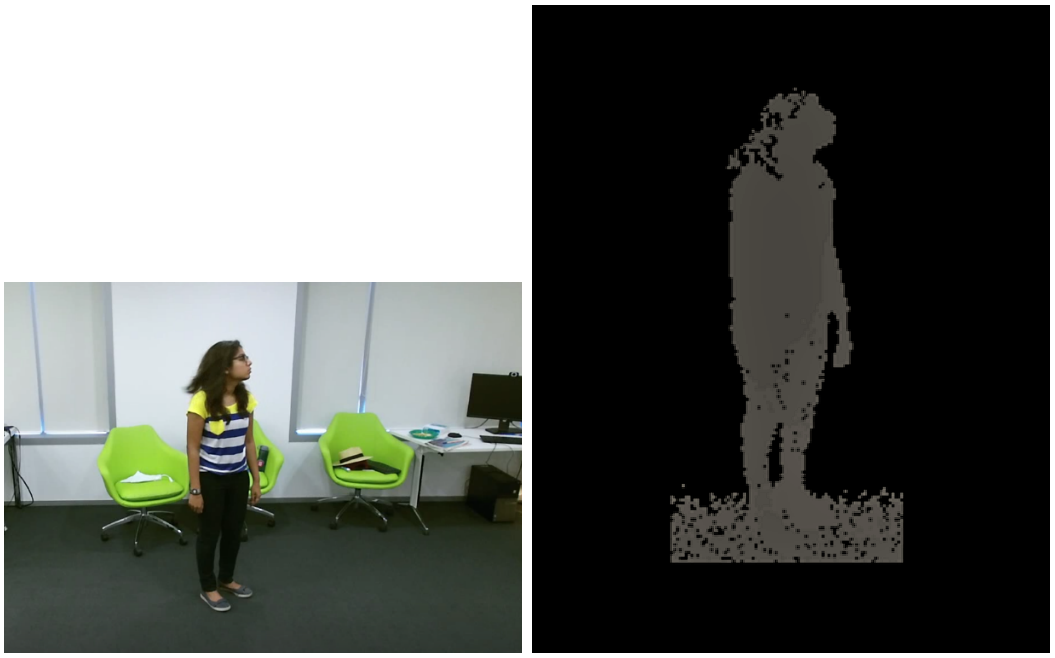


Figure 5.4: Moving head back and forth.

There are the 60 classes in NTU RGB+D, which we can split into two groups: single person action, and two person action.

Table 5.1: NTU RGB+D 60 Action Classes.

Person	**Group**	**Classes**
one	food	drink water eat meal/snack
one	refresh	brushing teeth brushing hair wipe face use a fan (with hand or paper)/feeling warm cheer up rub two hands together
one	movement	drop pickup throw sitting down standing up (from sitting position) hopping (one foot jumping) jump up kicking something
one	study	reading writing tear up paper
one	unwell	sneeze/cough

Person	Group	Classes
		staggering falling nausea or vomiting condition
one	device	make a phone call/answer phone playing with phone/tablet typing on a keyboard check time (from watch) taking a selfie
one	gesture	touch head (headache) touch chest (stomachache/heart pain) touch back (backache) touch neck (neckache) nod head/bow shake head hand waving cross hands in front (say stop) pointing to something with finger put the palms together clapping salute
one	clothes	wear jacket

Person	**Group**	**Classes**
		take off jacket
		wear a shoe
		take off a shoe
		wear on glasses
		take off glasses
		put on a hat/cap
		take off a hat/cap
		put something inside pocket / take out something from pocket
two	animosity	punching/slapping other person
		kicking other person
		pushing other person
		point finger at the other person
		touch other person's pocket
		walking apart from each other
two	friendship	hugging other person
		pat on back of other person
		giving something to other person
		handshaking
		walking towards each other

Although this currently is the largest RGB-D human activity dataset, it suffers from the following issues:

1. Background variation is very limited.
2. Almost all activities are in the middle of the screen.
3. Lighting variation is limited.
4. The activities are acted and not natural.
5. only Activities are indoor only.
6. Clip duration is short.

Our research started with this dataset, giving us the opportunity to compare RGB versus depth, and to experiment with multi-model architectures. Also, despite its large size, we we were able train from scratch a deep learning model, and transfer its to other smaller datasets.

5.2 Human3.6M

Human3.6m contains 3.6 million 3D human poses, each with skeleton, RGB image, and depth map. Eleven professional actors (6 males and 5 females) performed 17 everyday activities. The videos are captured at high resolution 50fps from 4 separated calibrated cameras. In addition, 10 MoCap cameras are used, as well as a 3D laser body scanner, to capture an accurate 3D volumetric models for each of the participant actors. Figure 5.5 shows room settings and camera locations used to capture the Human3.6m dataset.

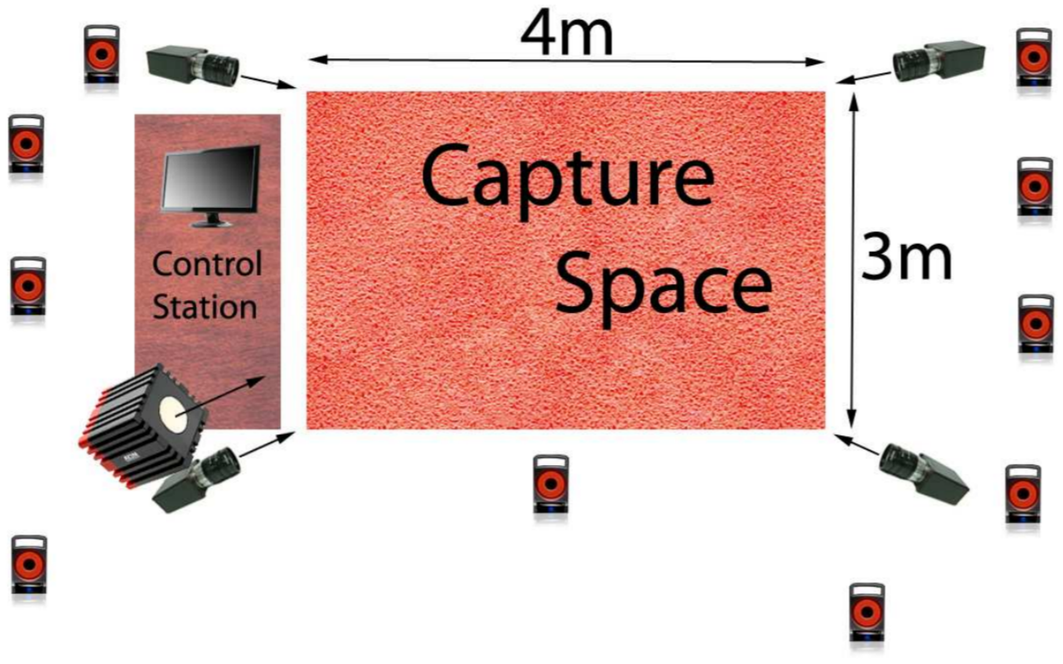


Figure 5.5: Camera and room settings from Ionescu et al. 2014. The large object is the laser scanner, the four high resolution camers are at the corners of the room, and the ten smaller MoCap cameras are scattered.

Here the list of activities classes in Human3.6m, grouped:

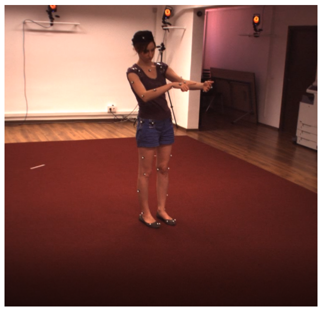
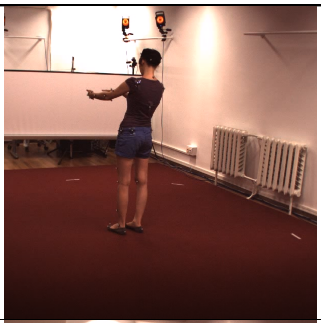
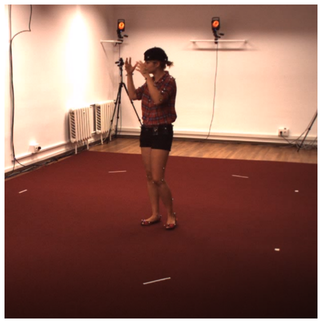
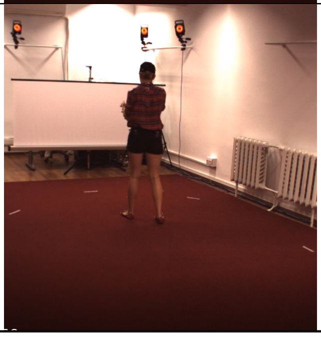
Table 5.2: Human3.6m Action Classes.

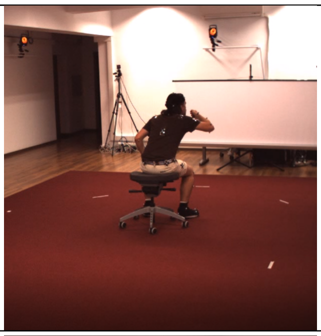
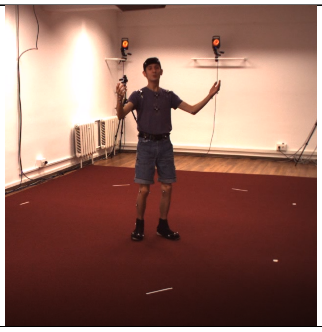
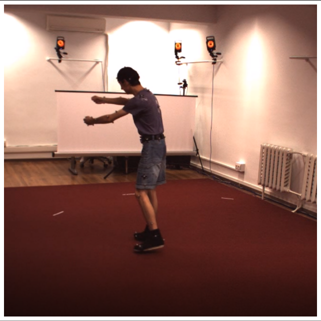
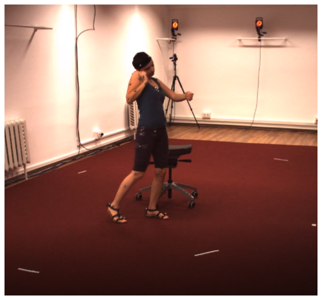
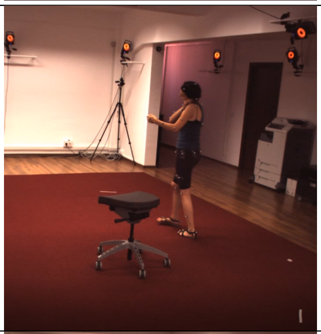
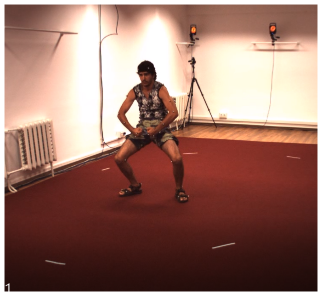
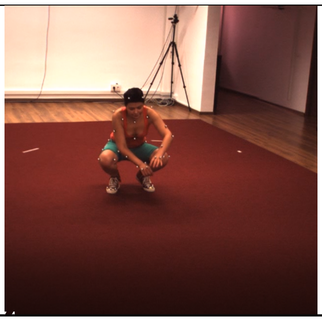
Group	**Classes**
Upper body movement	Directions
	Discussion
Full body upright variations	Greeting
	Posing
	Purchases
	Taking Photo
	Waiting

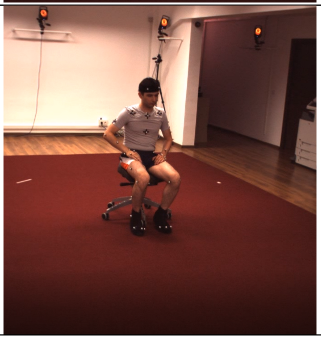
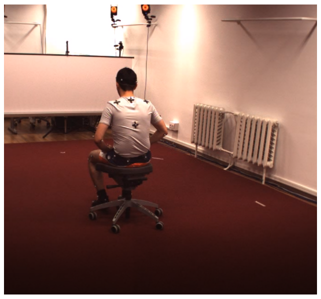
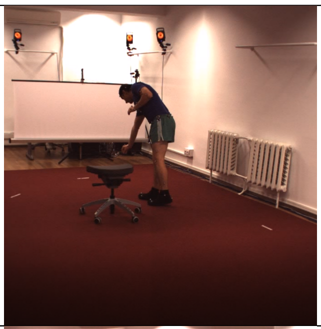
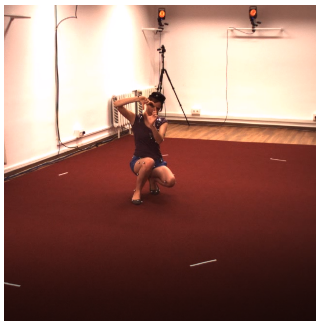
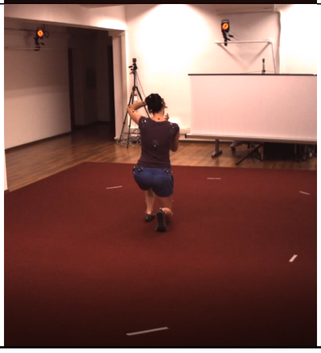
Group	**Classes**
Walking variations	Walking Walking Dog Walking Pair
Variations while seated on a chair	Eating Phone Talk Sitting Smoking
Sitting on the floor	Sitting Down
Various Movements	Miscellaneous

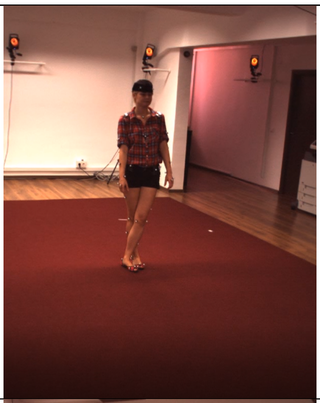
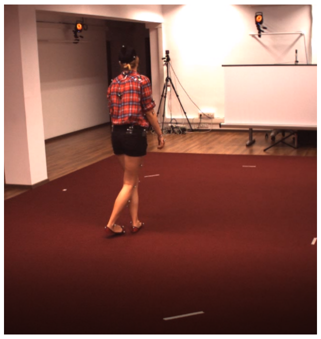
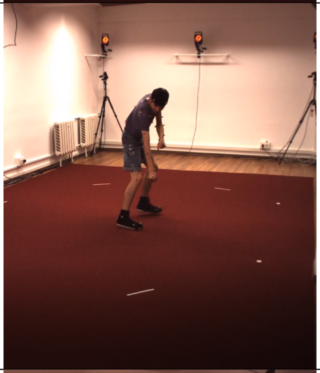
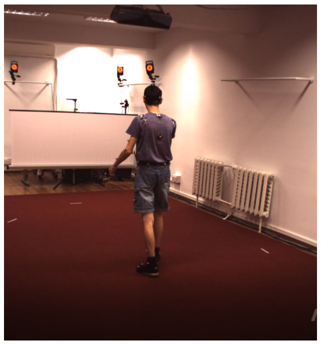
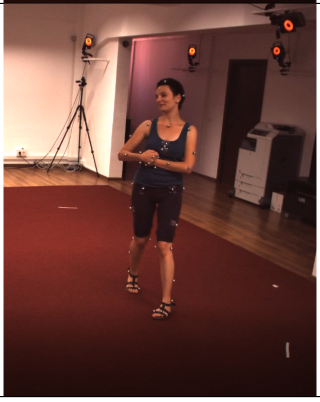
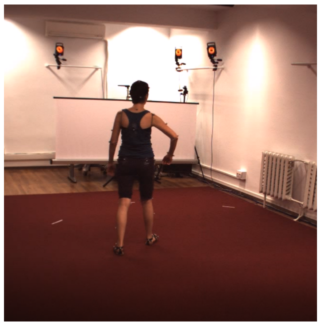
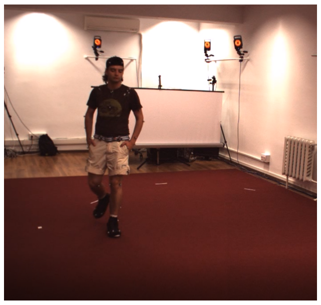
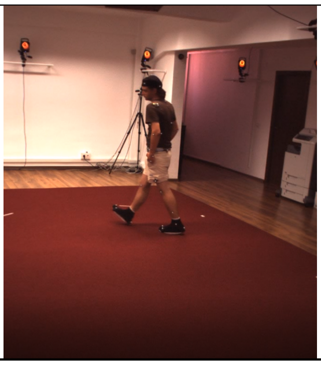
Here are examples of the classes,under two different views.

Table 5.3: Sample frames from Human3.6m

Classes	**Front view**	**Back view**
Directions		
Discussion		

Classes	Front view	Back view
Eating		
Greeting		
Phone Talk		
Posing		
Purchases		

Classes	**Front view**	**Back view**
Sitting Down		
Sitting		
Smoking		
Taking Photo		

Classes	Front view	Back view
Waiting		
Walking Dog		
Walking Pair		
Walking		

Although this currently is the largest MoCap human activity dataset, it suffers from the following issues:

1. Background variation is very limited.
2. Almost all activities are in the middle of the screen.
3. Lighting variation is limited.
4. The activities are acted and not natural.
5. Activities are indoor only.
6. Clip duration is long, but has only one label per clip.

5.3 Critique

The NTU RGB+D and Human3.6m datasets are currently the largest labeled datasets with depth information. These enabled us to train deep neural networks from scratch, and to experiment with different algorithms. However, both datasets still suffer with nearly identical data limitations, which are due mainly to sensor limitations.

6

Probabilistic human motion prediction

In this chapter, we design and demonstrate a novel sequence-to-sequence model for probabilistic human motion prediction, trained with an improved version of GAN, in which we use a custom loss function designed for human motion prediction. This work is an improvement over our previous HP-GAN work Barsoum, Kender, and Liu 2018 which was based on WGAN-GP Gulrajani et al. 2017.

Our model learns a probability density function of future human poses conditioned on previous poses. It predicts multiple sequences of possible future human poses, each from the same input sequence but using a different vector z drawn from a random distribution. Furthermore, to quantify the quality of the non-deterministic predictions, we simultaneously train a motion quality assessment model that learns the probability that a given skeleton sequence is a real human motion.

Accurate short-term (several second) predictions of what will happen in the world given past events is a fundamental and useful human ability. Such aptitude is vital for daily activities, social interactions, and ultimately survival. For example, driving requires predicting other cars' and pedestrians' motions in order to avoid an accident;

handshaking requires predicting the location of the other person's hand; and playing sports requires predicting other players' reactions. In order to create a machine that can interact seamlessly with the world, it needs a similar ability to understand the dynamics of the human world, and to predict probable futures based on learned history and the immediate present.

However, the future is not deterministic, so predicting the future cannot be deterministic, except in the very short term. As the predictions extend further into the future, uncertainty becomes higher. People walking may turn or fall; people throwing a ball may drop it instead. Some predictions are more plausible than others, and have a higher probability. Our work generates and quantifies these differences.

6.1 Design contributions

We have created a model that can predict multiple plausible future human (skeleton) poses from a given past. The number of poses taken from the immediate past, and the predicted number of poses in the future, which can be unrestricted, are parameters to the model. To accomplish this, we use a modified version of a GAN, with a custom loss function that takes into consideration human motion and human anatomy.

The generator is an adaptation of a sequence-to-sequence model Sutskever, Vinyals, and Le 2014 of poses derived from a RNN. The quality assessment network is a multilayer LSTM network with temporal attention. The discriminator is a multi-layer dense network with multiple skip connections. We use the discriminator network to train the generator, and we use the quality network to distinguish between a real sequence of poses from a fake one. Even though a single discriminator could assess the quality of the network, we train a separate network for quality assessment in order to avoid collusion from the generator. In effect, this additional network independently extends the ground truth labeling.

We train our model on all actions at once, so its output is not conditioned on any specific human action. Our model takes as input a sequence of previous skeletal poses, plus a random vector z, from the reduced sequence space, which samples possible future poses. For each such z value, the model generates a different output sequence of possible future poses.

We use a RNN for the generator because RNNs are a class of neural networks designed to model sequences, especially variable length sequences. They have been successfully used in machine translation Sutskever, Vinyals, and Le 2014, caption generation from images Donahue et al. 2015; Jia et al. 2015, video classification and action recognition Baccouche et al. 2011; Donahue et al. 2015; Ng et al. 2015a, action detection Yeung et al. 2016, video description Donahue et al. 2015; Kiros, Salakhutdinov, and Zemel 2014; Pan et al. 2016; Yao et al. 2015b, sequence prediction Graves 2013; Srivastava, Mansimov, and Salakhutdinov 2015 and many others.

We structure the learning by using a GAN because GANs Goodfellow et al. 2014 are a class of unsupervised learning algorithms, inspired by game theory Sion 1958, which allow the generation of futures that are not tied to specific ground truth. Among other domains, they have been used to generate impressive realistic images.

A GAN is based on two networks that compete against each other. A generator network tries to create random samples that are indistinguishable from training samples, and a discriminator (or critic) network tries to distinguish the generated sample from the training ones.

However, GANs have several weaknesses. They can be difficult to train, and unstable in their learning. Their loss value does not necessarily indicate the quality of the generated sample. The training can collapse easily. Recent literature Arjovsky, Chintala, and Bottou 2017; Gulrajani et al. 2017; Mao et al. 2016; Neyshabur, Bhojanapalli, and Chakrabarti 2017; Qi 2017; Uehara et al. 2016 tries to improve GAN training and provide a theoretical guarantee for its convergence. In our work, we address this by adding a custom loss based on the skeleton physics in addition to the GAN loss, in order to stabilize and improve the training. We have also borrowed some ideas from the improved Wasserstein GANs Gulrajani et al. 2017, which improved the stability in most of the GAN-based systems that we experimented with.

To quantitatively assess the quality of the non-deterministic predictions, we simultaneously and independently train a novel motion quality assessment model, which learns the probability that a given skeleton sequence is a real human motion. We use the outcome of such assessment in order to know which models provide the best predictions.

We then test our motion prediction model on two large datasets each captured with a different modality. The first is the NTURGB-D Shahroudy et al. 2016a dataset, which is the largest available RGB-D and skeleton-based dataset, with data captured by using the Microsoft Kinect v2 sensor. The second is the Human3.6M Ionescu, Li, and Sminchisescu 2011; Ionescu et al. 2014 dataset, which is one of the largest available datasets derived from motion capture (MoCap) data.

6.2 Motion prediction and assessment

In this section we more closely examine the state of the art in human motion prediction and in human motion quality assessment.

Since the introduction of the Kinect sensor, there has been much work on recognizing human action and predicting human poses from skeleton data. For example, predicting human poses conditioned on previous poses using deep RNNs Fragkiadaki et al. 2015; Jain et al. 2016; Martinez, Black, and Romero 2017 is due in part to this availability of large human motion datasets Ionescu, Li, and Sminchisescu 2011; Ionescu et al. 2014; Shahroudy et al. 2016a.

6.2.1 Human motion prediction

In general, human motion prediction can be categorized into two categories: probabilistic and deterministic prediction.

6.2.1.1 Probabilistic prediction

Most work in probabilistic human motion prediction uses non-deep learning approaches Fragkiadaki et al. 2015; Koppula and Saxena 2013; Lehrmann, Gehler, and Nowozin 2014; Pavlovic, Rehg, and MacCormick 2000; Sidenbladh, Black, and Sigal 2002; Wang, Fleet, and Hertzmann 2005, 2008.

In Lehrmann, Gehler, and Nowozin 2014, the authors propose simple Markov models that model observed data, and use the proposed model for action recognition and task completion. The limitation in this approach is that it predicts motion from a single action only, and assumes that human motion satisfies the Markov assumption. In Wang, Fleet, and Hertzmann 2008, the authors introduce Gaussian process dynamical models (GPDMs) to model human pose and motion. However, they train their model on each action separately, and constrain the model to a Gaussian process.

In Sidenbladh, Black, and Sigal 2002, the authors map human motion to a low dimensional space, and use the coordinates as an index into a binary tree that predicts the next pose, thus casting the prediction problem into a search problem. However, this approach can not generalize to previously unseen motions.

In Pavlovic, Rehg, and MacCormick 2000, the authors use switching linear dynamic systems learned through a Bayesian network, and in Koppula and Saxena 2013 the author used conditional random fields (CRF) to model spatio-temporal dynamics.

In contrast to all the above, our work does not use any statistical models to constrain the motion. As far as we know, we are the first to use deep neural networks for probabilistic motion prediction.

6.2.1.2 Deterministic prediction

Recent human motion prediction, which relies on deep RNNs Fragkiadaki et al. 2015; Jain et al. 2016; Martinez, Black, and Romero 2017 or deep neutral networks Bütepage et al. 2017; Li et al. 2018, is primarily deterministic.

In Fragkiadaki et al. 2015, the authors mix both deterministic and probabilistic human motion predictions. Their deterministic aspect is based on a modified RNN called Encoder-Recurrent-Decoder (ERD). It adds fully connected layers before and after a LSTM Hochreiter and Schmidhuber 1997 layer, and minimizes an Euclidean loss. Their probabilistic aspect uses a Gaussian Mixture Model (GMM) with five mixture components, and it minimizes the GMM negative log-likelihood. For both aspects, they predict a single future human pose at a time. To predict more, they recurrently feed the single predicted pose back to the input. However, one drawback of this approach is error drifting, where the prediction error of the current pose will propagate into the next pose. In contrast, we predict multiple human poses at once, thus avoiding error drifting. In addition, we do not impose any statistical model constraints like GMMs over the motions.

In Jain et al. 2016, the authors develop a general framework that converts a structure graph to an RNN, called a Structure-RNN (S-RNN). They test their framework on different problem sets including human motion prediction, showing that it outperforms the current state of the art. However, they need to design the structure graph manually and task-specifically.

In Martinez, Black, and Romero 2017, the authors examine recent deep RNN methods for human motion prediction, and show that they achieve start-of-the-art results with a simpler model, by proposing three simple changes to RNN: a sequence-to-sequence architecture, sample-based loss, and the use of residual connections with the RNN.

In Bütepage et al. 2017, the authors use an encoder-decoder network based on a feed-forward network, and compare the results of three different such architectures: symmetric, time-scale, and hierarchical.

However, the main issues of deterministic prediction of human motion are two-fold. The future is not deterministic, so the same previous poses could lead to multiple possible poses. And, using an L_2 norm can cause the model to average between two possible futures Mathieu, Couprie, and LeCun 2015, resulting in blurred motions. Furthermore, training a deterministic motion predictor on a specific dataset will cause the model to learn human poses from that dataset only, which will limit its generalization.

6.2.2 Non-human prediction

The prediction of multiple possible futures using RNNs has precedents. In Graves 2013, the authors use an LSTM Hochreiter and Schmidhuber 1997 to generate text and handwriting from an input sequence. They generate one item at a time, by sampling the resultant probability. Then, they append the predicted item to the input sequence and remove its first item, and iterate. This creates a sequence of desired length, but the method may eventually create an input to the LSTM that does not contain any of the original input. In contrast, we pursue a method that trains our model to generate the entire desired output sequence of poses all at once.

The prediction of single or multiple possible futures using GANs also has precedents. In Mathieu, Couprie, and LeCun 2015, the authors trained a convolution model for both the generator and the discriminator in order to predict future frames. They improved the predictions by adding an image gradient difference loss to the adversary loss. However, they again only predict a single possible future and, due to the use of the convolution network, the model can only predict a fixed length output. In contrast, we support variable length input and variable length output, and can also generate multiple possible futures from the same input.

In Chen et al. 2017, the authors predict or imagine multiple frames from a single image. They generate affine transformations between each frame, and apply them to the original input image to produce their prediction. Although this can imagine multiple futures from the same input image, a single image is not sufficient to capture the temporal dynamics of a scene. Furthermore, it makes the oversimplified assumption that the change in object appearence between the images can be captured by using an image-based 2D affine transform.

6.2.3 Quality assessment

Compared to the amount of work on motion editing and synthesis in computer graphics, the research on automatic motion quality evaluation has received little attention Harrison, Rensink, and Panne 2004; Hodgins, O'Brien, and Tumblin 1998; Reitsma and Pollard 2003; Ren et al. 2005. The existing techniques were typically designed for special types of motions, or for motions obtained from software used to edit character motion Wang, Chen, and Wang 2014. A novel aspect of our motion quality assessment model is that it is trained simultaneously with the motion prediction model.

6.3 Design details

In the human motion prediction problem, the system takes a sequence of human poses as input, and predicts multiple valid future poses.

Let $x = \{x_1, x_2, ..., x_m\}$ be the input sequence of human poses and $y = \{y_1, y_2, ..., y_n\}$ be the predicted sequence, where the concatenated sequence of x and y corresponds to a single activity. Each pose can be presented with 3D joint locations or joint angles. Since the future is not deterministic, our goal is to be able to sample multiple possible future y from the same input sequence x. By that we mean that the model should learn an implicit conditional probability of the future sequence, conditioned on the input sequence $P(y|x)$.

Our approach is based on a GAN architecture as shown in Figure 6.1. The discriminator distinguishes between valid and not valid human poses. The generator creates future human poses based on an input sequence and a z vector that samples different output sequences. The quality network independently learns the probability that a sequence of poses are valid human poses. (We defend this additional network below.) We have augment with multiple additional losses the standard GAN, in order to stabilize the training, to control different aspect of the predicted human motion, and to improve the quality of the predicted poses.

We demonstrate the power of the features learned by the discriminator, by showing how it can be used on different but related tasks. We transfer the discriminator's learning to an action recognition task, and show that it speeds up the training by more than a factor of 2, compared to training from scratch.

The generator, as shown in Figure 6.3, is a modified version of the sequence-to-sequence Sutskever, Vinyals, and Le 2014 network. It takes as input a sequence of human poses, plus a z vector drawn from a uniform or Gaussian distribution $z \sim p_z$. The drawn z value is then mapped to the same space as the output states of the

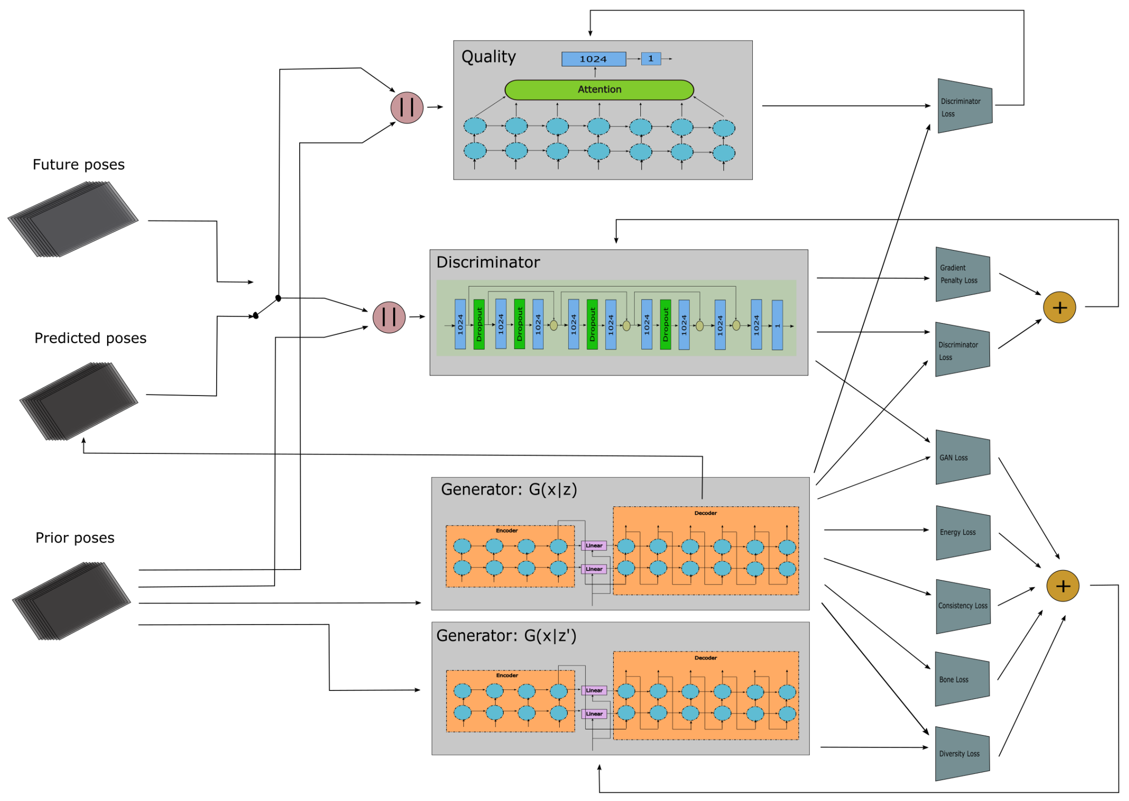


Figure 6.1: Overall architecture of our prediction model. The discriminator is a feed-forward network with skip connections, which takes the concatenation of the input sequence and the predicted sequence, and classifies the concatenation as a real or fake human motion. The two generator networks each take a sequence of human poses as input plus a z vector, and generate a sequence of future human poses, where each z value samples a different future. The second generator is governed by a diversity loss, which controls the variety of the predictions produced. The quality network is a recurrent network with temporal attention, which computes the probability that the concatenation sequence is human. Note that the quality network does not affect the generator.

encoder. We then simply add the mapped value of z to the encoder states, and use the resulting state as the initial state of the decoder. We map z to the output of each layer in the encoder, and then feed the last output of the encoder to the first input of the decoder. We use GRU Chung et al. 2014 for our sequence-to-sequence network. We have also tried LSTM Hochreiter and Schmidhuber 1997, but we did not notice any quality difference.

Let G be the network shown in figure 6.3. We have $y = G(x, z; \theta_g)$, where θ_g are the network parameters that we need to learn. Each drawn value of z will sample different valid future poses from the given input x.

6.3.1 Improving on the GAN

Most recent human pose predictions using deep RNNs Fragkiadaki et al. 2015; Martinez, Black, and Romero 2017 treat human motion prediction as a regression problem. However, solving human motion prediction using regression suffers from two deficiencies. First, it learns one outcome at a time, and as the predicted sequence length increases, this outcome becomes less probable. Second, using the usual ℓ_1 or ℓ_2 loss creates artifacts Mathieu, Couprie, and LeCun 2015, such as predicting the average of multiple possible outcomes which is not the correct prediction.

Instead, we use an adversarial training scheme for three reasons. First, it allows the generation of multiple futures from a single past. Second, it allows the generator to be trained without explicitly using the ground truth of real futures. Third, it implicitly learns the cost function for the prediction based on the data.

Generative adversarial networks (GAN) was introduced by Goodfellow et al. 2014. It is a unsupervised learning technique inspired by the minimax theorem Sion 1958, in which the generator network and the discriminator network try to outdo each other. The training itself alternates between both networks. In the original paper, the generator learns to generate images close to real images, and the discriminator learns to distinguish between the generated image and the real image from the dataset. In the final steady state, the trained discriminator should predict if an image from the generator network is real or not with equal 50% probabilities.

However, the original GAN algorithm comes with some drawbacks. Is not stable and is difficult to train, because of its use of Jensen-Shannon (JS) divergence as its loss function. JS can result in zero divergences if two probabilities completely overlap, which can then lead to vanishing gradients in the discriminator network.

WGAN Arjovsky, Chintala, and Bottou 2017 replaces the JS distance with the Earth Mover Distance (EMD), which is defined and continuous almost everywhere. And according to the author, this mitigates the need to carefully maintain a balance between training the discriminator versus the generator. The discriminator in WGAN does not output a probability, and it does not discriminate between synthetic input and real input–which is why the author renamed the discriminator network as a "critic" network.

Nevertheless, WGAN does not address all the concerns, since the critic still must maintain a Lipschitz constraint. In order to do so, the author clips the weights of the network. But this adversely affects the quality of the generator. To address this further problem, the variant WGAN-GP algorithm Gulrajani et al. 2017 replaces weight clipping in the generator with an added penalty to the loss in the critic, which is based on the computed norm of the gradient with respect to the critic input.

We were able to verify these WGAN-GP improvements in our domain of human pose prediction. And, our prior HP-GAN work Barsoum, Kender, and Liu 2018 used a modified version of WGAN-GP in the training, in order to help ensure convergence and to stabilize the training. However, we found through experimentation that we can make a standard GAN as stable as a WGAN-GP by simply borrowing the penalty loss from WGAN-GP. So, the path of our research has explored GAN, then WGAN, then WGAN-GP, but ends up with what might be called GAN-GP.

There are further reasons to prefer our modified GAN instead of a WGAN-GP. First, a GAN discriminator computes a probability value that indicates if the input is real or fake, whereas a WGAN and its family use critic that computes a floating point number which cannot be interpreted. Second, the loss value in GAN is positive and decreases when the network converges, whereas in WGAN it is difficult to tell if the network converges or not from the loss alone, without looking at the actual generated poses. And third, in our experiments with GAN, we have noted that if we continue training after convergence, the convergence continues, whereas with WGAN, it can begin to diverge.

6.3.2 Human pose prediction: architecture

We describe the architecture further.

Our overall model is based on a GAN Goodfellow et al. 2014 augmented with the gradient penalty borrowed from WGAN-GP Gulrajani et al. 2017, together with and losses specific to human poses. In Figure 6.1, ”Future poses” refers to the ground truth future poses from the dataset, and ”Prior poses” refers to their corresponding ground truth previous poses.

6.3.2.1 Discriminator network

The discriminator network, shown in more detail in Figure 6.2, is a fully connected feed-forward network with skip connections. It outputs a single probability value between 0 and 1, with 0 meaning that the input is fake (i.e., generated), and 1 meaning that the input is real (i.e., taken from the ground truth dataset).

The input to the discriminator is the concatenation of the prior sequence of human poses, either with the future sequence of human poses from the ground truth, or with the predicted sequence of human poses from the generator network. The job of the discriminator is to compute the probability if a given sequence of human poses is real or fake. The second branch in the discriminator is for action classification; this branch is not used during GAN training.

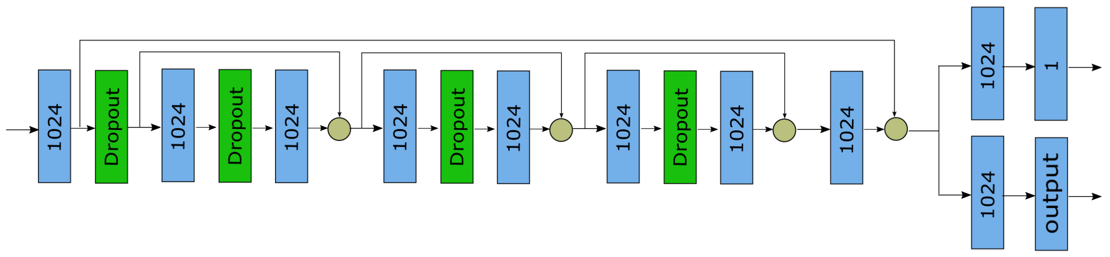


Figure 6.2: Discriminator network is a full connected multi-layer network with a skip connectio. It ends with two branches, top branch for GAN training and bottom branch for activity recognition training.

In addition to its usual role in a GAN, we also use the discriminator network for activity recognition. We show that the features learned by the discriminator during its GAN training help speed up the classification task.

6.3.2.2 Generator network

The generator, shown in more detail in Figure 6.3, is a sequence-to-sequence network. It takes as input previous human poses and a z vector, and produces a sequence of predicted future human poses. The z vector is a 128-dimensional float vector drawn from a uniform or Gaussian probability distribution.

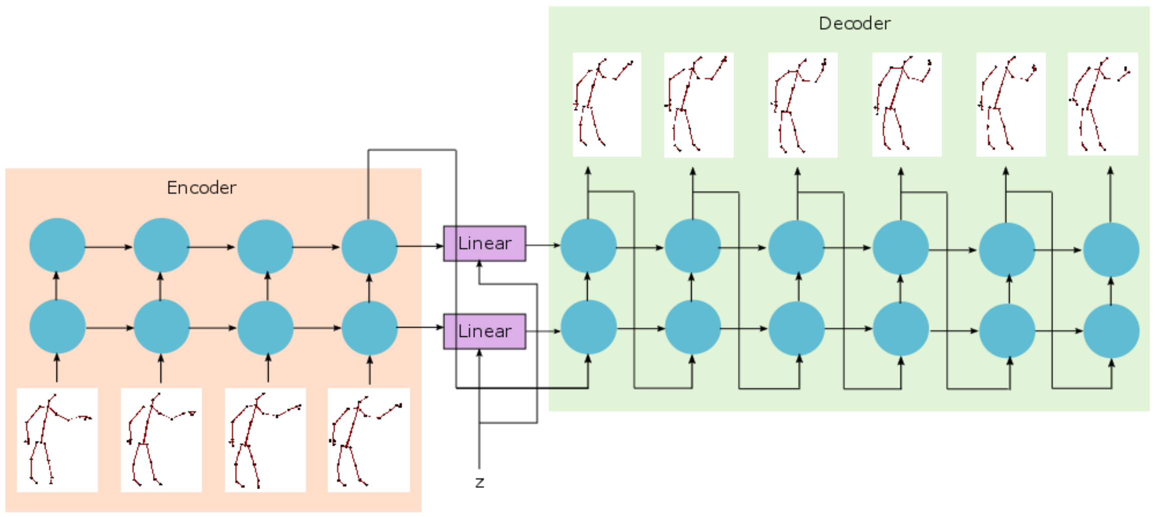


Figure 6.3: Generator network is a modified sequence-to-sequence network.

6.3.2.3 Quality network

The quality network, shown in more detail in Figure 6.4, is a multi-layer recurrent network with temporal attention. The attention layer's purpose is to learn which part of the sequence is more important for generating the probability of the input sequence of poses. Instead of using the more common methods of either averaging the outputs across time, or using the last output in the recurrent network, the attention layer learns from the training data the importance of each output and captures this information in a vector.

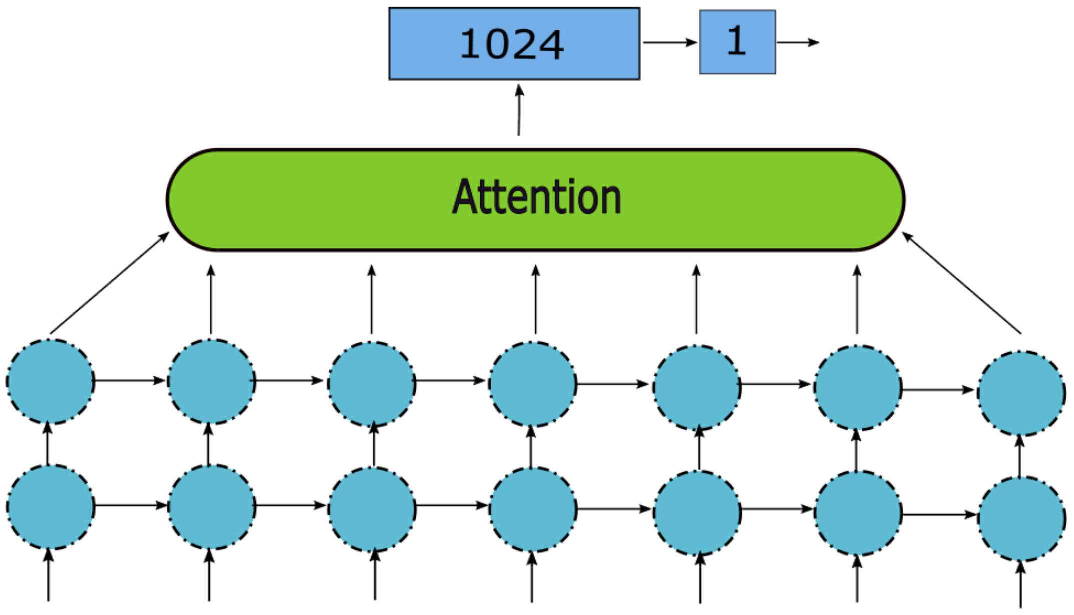


Figure 6.4: The quality network generates a measure to quantify the quality of the generated poses. It is a GRU recurrent network with attention.

In more detail: Let the learned weights be α_t, the output of the network be y_t and the attention final output be α, where t is the time axis. Then the weights of the attention and the final output are computed in the usual way, respectively, as $\alpha_t = Softmax(U^T y_t) = \frac{e^{U^T y_t}}{\sum_\tau e^{U^T y_\tau}}$ and $\alpha = \sum_\tau \alpha_\tau y_\tau$, where U is the learned parameter of the attention layer.

6.3.3 Human pose prediction: losses

Prior poses from the ground truth and future poses from the ground truth are concatenated together to form a real pose sequence. Similarly, prior poses from the ground truth and generated poses from the generator are concatenated together to form a fake sequence. Both real and fake sequences are used by the discriminator and the quality network to compute their loses. These and other GAN losses used to improve all three network types, by cycling through discriminator loss, generator loss, and quality loss.

6.3.3.1 Discriminator loss

Based on numerous experiments, we found that almost all GAN algorithms were stabilized by adding a gradient penalty to the discriminator loss, similar to the one defined in WGAN-GP. We are not the first to discover this Fedus et al. 2017, but we are the first to show it on a complex problem such as skeleton pose prediction. The resulting stabilization was also a very useful research tool, as it enabled us to try multiple GAN algorithm variations.

The discriminator loss for human prediction is defined from three parts as follows:

$$L_d = L^d_{gan} + \lambda_{gp} L^d_{gp} + \alpha L^d_2 \tag{6.1}$$

The first part, L^d_{gan}, is the original GAN discriminator loss, defined as:

$$L^d_{gan} = log(D(x||y)) + log(1 - D(x||G(x,z))) \tag{6.2}$$

where $||$ indicates concatenation, x is the input sequence, y is the future sequence, and z is a random vector drawn from a uniform distribution.

The second part, L_{gp}, is the gradient penalty loss, defined as:

$$L_{gp} = (\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2 \tag{6.3}$$

where $\hat{x} = \epsilon(x||y) + (1-\epsilon)(x||G(x,z))$ and $\epsilon \sim U[0,1]$.

The third part, L_2^d, is the standard L_2^d regularization, defined as:

$$L_2^d = \|\theta_d\|_2 \tag{6.4}$$

In all of our experiments, we set $\lambda_{gp} = 10$ and $\alpha = 0.001$.

6.3.3.2 Generator loss

For the generator loss, we use the weighted combination of five loss components. The first loss is the standard GAN adversary loss, which is defined in Goodfellow et al. 2014. The remaining losses are unique contributions of this work.

The second loss is a pose gradient loss, which minimizes the delta between two consecutive poses and helps to ensure temporal pose consistency. The third loss is a diversity loss, which controls the variety between each sequence of poses generated from the same inputs but different z values. These second and third losses were found to be necessary for smoothness and novelty, respectively. They should be easy to generalize to other tasks involving articulated real world objects subject to inertial constraints.

The fourth loss is an energy loss, which minimizes an approximate calculation of the physical energy that would be necessary to generate the human's physical motion, inspired from work in the animation of human motion. The fifth loss is a bone loss, which minimize the delta between the bone lengths between the predicted skeleton and the prior skeleton. These fourth and fifth losses both contribute to realism. They should be easy to generalize to other tasks involving involving energy-minimizing objects which are constructed of rigid components.

Taken together, the four additional generator losses ensure smooth diverse motion sequences that realistically capture the grace and stability of the moving human form.

The generator network total loss L_g is therefore defined as:

$$L_g = L_{gan}^g + \alpha_{pg} L_{pg}^g + \alpha_d L_d^g + \alpha_e L_e^g + \alpha_b L_b^g \tag{6.5}$$

The definition of the five component generator losses follow.

Standard GAN adversary loss L_{gan}^g is defined as:

$$L_{gan}^g = -log(D(x||G(x,z))) \tag{6.6}$$

Pose gradient loss L_{pg} is defined as:

$$L_{pg}^g = \|\nabla_t y\|_p = \left[\sum_t |y_t - y_{t-1}|^p\right]^{1/p} \tag{6.7}$$

where $\|\nabla_t y\|_p$ computes the gradient over time for the predicted sequence. Although this is a generalized Minkowski distance, we use $p = 2$ (Euclidean) in our training, in keeping with the physical nature of the problem. This loss helps to reduce the delta between skeleton poses between each consecutive frame, since for valid human motions captured in our datasets, this displacement between frames is small.

We found that L_{pg}^g is critical during training. If α_{pg} is too large, the effect of L_{pg}

loss becomes too high and we obtain less displacement between poses across time. In some cases this can cause the copying of the same pose. Conversely, we have found that L_{pg} is not critical if we train on an entire large dataset, like all the 49 classes of NTURGB-D, where motions vary. However, for training small subsets of one or two classes, then its value becomes critical in avoiding motion discontinuities at the first predicted human pose. Furthermore, to make sure that L^g_{pg} loss does not reach zero, we set a minimum value for the loss as $L^g_{pg} = max(C, \|\nabla_t y\|_p)$, where C is a hyper parameter, to avoid pushing two consecutive human poses to match exactly.

Diversity loss L^g_d is defined as:

$$L^g_d = 1 - \frac{1}{1 + e^-\eta|G(x, z_1) - G(x, z_2)|} \tag{6.8}$$

This controls the variation in the predicted poses, given the same input of poses but different randomizing z values. Therefore, it encourages the trainer to increase the absolute delta $|G(x, z_1) - G(x, z_2)|$ with different z values.

The structure of this space is of such high dimension that manually ensuring diversity is impractical. But without such a loss, we found that many of the generated sequences are not noticeably different to the human experimenter, regardless of the input z value.

Energy loss L^g_e is defined as:

$$L^g_e = \sum_t [\beta_v v(t)^2 + \beta_a a(t)^2] \tag{6.9}$$

The above equation is based on the energy expenditure estimations defined for exergaming applications in Nathan et al. 2015. The velocity v and acceleration a are computed from the center of mass in each skeleton pose. For natural human motion, the energy expenditure generated by motion needs to be minimized. A similar technique is observed in motion animations.

We note that the original equation had two separate energy terms, one for kinetic energy based on velocity, and another for potential energy based on height (since many exergames involve jumping). However, using the original equation degraded the prediction quality. We found that minimizing energy tended to minimize skeleton heights, and that it generated poses that tended to shrink over time. Instead, in our revision, minimizing velocity tends to reduce drift, and minimizing acceleration tends to reduce jitter.

Bone loss L_b^g is defined as:

$$L_b^g = \sum_t \left[\sum_i |b_t^i - b_{gt}^i|^2 \right]^{1/2} \tag{6.10}$$

where b_{gt}^i is the ground truth bone length and b_t^i is the predicted bone length, both at time t. We iterate through all bones using index i, and sum over all the future skeleton poses using index t. Ideally, this value should be zero.

6.3.3.3 Quality loss

We use another network, shown in figure 6.4, to judge the quality of the predicted skeleton poses and to decide which model to choose.

The quality loss is the same loss used in GAN Goodfellow et al. 2014, with the exception that the generator does not use this loss in its training.

The quality loss is defined as:

$$L_q = L^q_{gan} + \alpha L^q_2 \tag{6.11}$$

where L^q_{gan} is the standard GAN loss defined as:

$$L^q_{gan} = log(Q(x||y)) + log(1 - Q(x||G(x,z))) \tag{6.12}$$

and L^q_2 is the same as the L^d_2 norm used by the discriminator network.

We use a separate network to measure the quality of the generator model, instead of using the discriminator network, because the discriminator is essentially part of the generator loss. Therefore, the generator is optimized to outperform the discriminator. The separate network allows a more independent measurement.

6.3.4 Human pose classification

To do classification, we use the pre-trained discriminator from the prediction training. We replace the last two layers with two new layers for classification, as shown in Figure 6.5. The two new layers are fully connected layers, and the last one has k outputs, where k is the number of classes used during training.

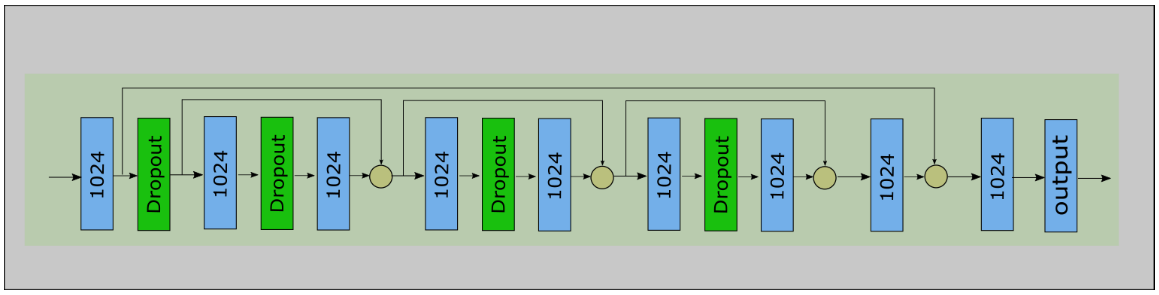


Figure 6.5: The discriminator network with the top two layers replaced.

Since the discriminator learned to differentiate between real and fake human motion, therefore it must have learned much of what constitutes a human motion. We postulated that the discriminator learns general features about human motion that can be used on other tasks, beyond computing the probability that a given sequence of poses are real or fake. In order to verify this assumption, we trained the classifier in Figure 6.5 twice, the first time with the discriminator weights, and a second time with random initialization. We observed that using the discriminator as the basis for the classifier training sped up the training, compared to training from scratch.

6.3.5 Training

For the training algorithm 1, we follow closely most GAN training methods. Inside the training loop, we iterate d times on the discriminator network, g times on the generator and q times on the quality network. We eventually used $d = 10$, $g = 2$ and $q = 1$. We tried different iteration values, and have tried to dynamically update the iteration count based on the losses of the discriminator and the generator, but none of those acceleration methods made any noticeable improvement. But in order to make the quality network training procedure more stable, we found we had to reduce its learning rate by half, compared to the discriminator and generator learning rate.

For all three networks, we used Adam Kingma and Ba 2014 to update network weights, and set the learning rate as $5e-5$ for the discriminator and generator network, and half of that for the quality network. For the supervised training, we trained the network shown in Figure 6.5 twice, one with the same weights as the discriminator and another with random initialization. The training was therefore done in two phases: first a usual GAN training using the losses described previously, on the discriminator, generator, and quality network, then an action classification training on the modified, discriminator network.

Here is the GAN training algorithm for the human pose prediction, which cycles through the three basic networks once per epoch:

Algorithm 1 Training HPGAN2 prediction

```
1: $d \Leftarrow 10$
2: $q \Leftarrow 1$
3: $g \Leftarrow 2$
4: $EPOCH \Leftarrow 0$ {Training loop}
5: while $EPOCH < MAX\_EPOCH$ do
6:   $d\_loss \Leftarrow 0$
7:   $q\_loss \Leftarrow 0$
8:   $g\_loss \Leftarrow 0$
9:   while $HasMoreData()$ do
10:    $past\_seq, full\_seq \Leftarrow NextBatch()$ {Get next minibatch}
11:    $z_1 \Leftarrow UniformRand(-1, 1)$ {Generate a random vector on each iteration}
       {Generate a second random vector on each iteration for the generator loss}
12:    $z_2 \Leftarrow UniformRand(-1, 1)$

       {Iterate $d$ times on the discriminator network}
13:    for $i \leftarrow 1$ to $d$ do
14:      $loss \Leftarrow ForwardDiscriminatorNetwork(past\_seq, full\_seq, z_1)$
15:      $UpdateDiscriminatorNetworkWeights(loss)$
16:      $d\_loss \Leftarrow d\_loss + loss$
17:    end for

       {Iterate $q$ times on the quality network}
18:    for $i \leftarrow 1$ to $q$ do
19:      $loss \Leftarrow ForwardQualityNetworkWeights(past\_seq, full\_seq, z_1)$
20:      $UpdateQualityNetworkWeights(loss)$
21:      $q\_loss \Leftarrow q\_loss + loss$
22:    end for

       {Iterate $g$ times on the generator network}
23:    for $i \leftarrow 1$ to $g$ do
24:      $loss \Leftarrow ForwardGeneratorNetworkWeights(past\_seq, full\_seq, z_1, z_2)$
25:      $UpdateGeneratorNetworkWeights(loss)$
26:      $g\_loss \Leftarrow g\_loss + loss$
27:    end for

28:    $EPOCH \Leftarrow EPOCH + 1$
29:  end while
30: end while
```

As shown in Algorithm 1, we iterate through each network separately. The generator network only knows about the discriminator network; it has no knowledge of the quality network. The reason we have both z_1 and z_2, instead of just one z, is to calculate the diversity loss; this requires two z values in order to increase the delta between the generator outputs of both z's.

Here the simpler human activity training algorithm:

Algorithm 2 Training human activity classification

```
1: $model\_new \Leftarrow NewDiscriminatorModel()$
2: $model\_finetune \Leftarrow LoadDiscriminatorModel()$

3: $EPOCH \Leftarrow 0$ {Training loop}
4: while $EPOCH < MAX\_EPOCH$ do
5:   while $HasMoreData()$ do
6:     $full\_seq, label \Leftarrow NextBatch()$ {Get next minibatch}

7:     $loss \Leftarrow ComputeLoss(model\_new, full\_seq, label)$
8:     $UpdateModel(model\_new, loss)$

9:     $loss \Leftarrow ComputeLoss(model\_finetune, full\_seq, label)$
10:    $UpdateModel(model\_finetune, loss)$

11:    $EPOCH \Leftarrow EPOCH + 1$
12:  end while
13: end while
```

As shown in Algorithm 2, we create two instances of the same discriminator model. One is initialized with random weights, and the other initialized with the best saved weights from the GAN training. The remainder of the training is a standard supervised training algorithm.

6.4 Summary

In this chapter, we discussed in detail our contribution to human motion estimation and understanding. The reason for the various components in our model, various losses and the training algorithm. In addition, to the architecture of the network. Next chapter, we will evaluate all those components on NTURGB-D and Human3.6m dataset.

7

Experimental results

In this chapter, we discuss the results of human action anticipation and recognition training on both the NTURGB-D and the Human3.6m datasets. We also show the effect of each part of the loss on the prediction outcome in the ablation study section. Moreover, we demonstrate the power of the learned human dynamics representation in the discriminator upon the action recognition problem, and show that it reaches the same quality in half the number of epochs compared to training from scratch.

7.1 Experimental setup

To verify our model capability, we have run multiple experiments on two of the largest human motion datasets: a Microsoft Kinect based dataset NTURGB-D Shahroudy et al. 2016a and a motion capture (MoCap) dataset Human3.6M Ionescu, Li, and Sminchisescu 2011; Ionescu et al. 2014.

The human poses in NTURGB-D dataset are based on skeleton data from the Kinect V2. The data is not perfect due to occlusions, people carrying objects, or people interacting with other persons. However, even with noisy skeletons,our model generalizes well on this dataset. The dataset consists of $56,880$ actions, and each action comes with the corresponding RGB video, depth map sequence, 3D skeletal data, and infrared video. We use only the 3D skeleton data. They contain the 3D locations of 25 major body joints at each frame, as defined by the Microsoft Kinect API. NTURGB-D has 60 action classes and 40 different subjects, and each action was recorded by three Kinects from different viewpoints. From those 60 classes, 49 classes are for single person actions, and the rest are for two people interacting together. Each clip is from 2 to 5 seconds long.

Human3.6M contains 3.6 million 3D human poses and their corresponding images, captured by a Vicon MoCap system. Each of these skeletons has 32 joints. The actions were performed by 11 professional actors covering 17 action classes. Each clip is about 4 minutes long. We used part of the data loader code from Martinez, Black, and Romero 2017.

For both datasets, we trained our model directly on the absolute 3D joint locations. We fed our model into a 3D point cloud, and from this training data the model learned the relationships between the joints in order to predict a valid human pose. This is more difficult than training on joint angles, which has fewer degrees of freedom. We train directly on the joint positions in order to use the same pipeline for both NTURGB-D and Human3.6m datasets, and in order to have a more generic model.

7.1.1 Pre-processing

For preprocessing, we computed the mean and standard deviation across the entire dataset. We then subtracted the mean from each joint, and divided the result by double the standard deviation, in order to reduce the number of joints close to the boundary range [-1, 1]. We tried other normalization techniques, for normalizing each joint to the range of [-1, 1], such as using the dimensions of the Kinect frustum at 5 meter depth (all activities were within 5 meters range to the sensor), or normalizing each joint by using min and max values in the dataset. Using mean and standard deviation helped align most activities to the center, and avoided the need to use camera intrinsic properties.

The two datasets have some important differences in structure, and we adjusted them to be closer to each other. The Human3.6M dataset has fewer clips than the NTURGB-D dataset; however each clip is much longer, by about two decimal orders of magnitude. In order to use the same pipeline for both datasets, we split Human3.6M clips into shorter segments, and we only used every other of its frames in our training, in order to better match the dataset frame rates. We also used 10000 epochs for the Human3.6M dataset, compared to 300 epochs for the NTU-RGBD dataset, due to its smaller number of clips. Also, for the Human3.6M dataset we selected a random subsegment during each iteration, and therefore each epoch often processed a different set of frames.

7.1.2 Interpreting the results

One of the main problems of a GAN is that the loss does not provide a reliable indication of the quality of the generated data. According to claims of the authors of WGAN Arjovsky, Chintala, and Bottou 2017 and WGAN-GP Gulrajani et al. 2017, one of the improvements that WGAN makes on the original GAN is that its loss value does in fact provide a quality measure. So, in our human motion prediction problem, this loss should provide some indication of how well the generated sequence looks like a valid human pose.

However, we have observed in our experiments that this loss is not strictly monotonic. A smaller loss does not always indicate a better quality in WGAN or WGAN-GP. Even worse, once the model reaches a good convergence state, further training often causes the model to diverge.

We found that instead of using WGAN or WGAN-GP, using a simple GAN but with a gradient penalty (a "GAN-GP") helped to mitigate this problem. With a GAN, the discriminator estimates a true probability which can be used to measure the quality of the estimated human motion. And with a GP, the loss value accurately and monotonically indicates the quality of the generator.

Nonetheless, our experiments compelled us to introduce another network to measure the quality of the predicted human poses. This is because the discriminator acts as a loss to the generator, and the generator is optimized to *reduce* the accuracy of the discriminator. Therefore, we added a quality network, whose sole purpose is to learn the probability that a given sequence is a valid human motion. The loss of the generator does not use this quality network.

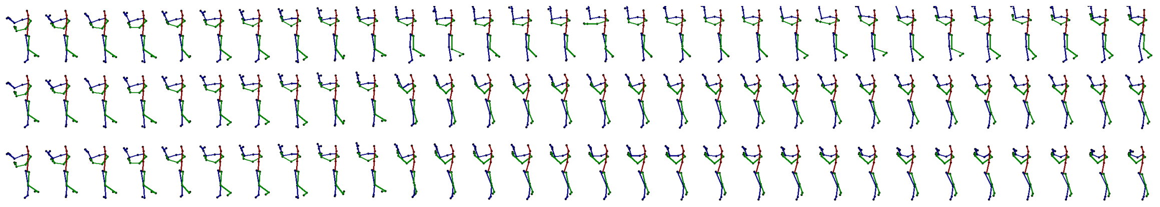
Therefore, in order to find the best model, inside the training loop we generate N predictions and compute the quality probability of each of them. We record the number of predictions k that have a probability of more than 50%. We keep track of the model with maximum k during training–but only after a certain number of warm-up epochs, in order to give the network a chance to learn how to evaluate human motions.

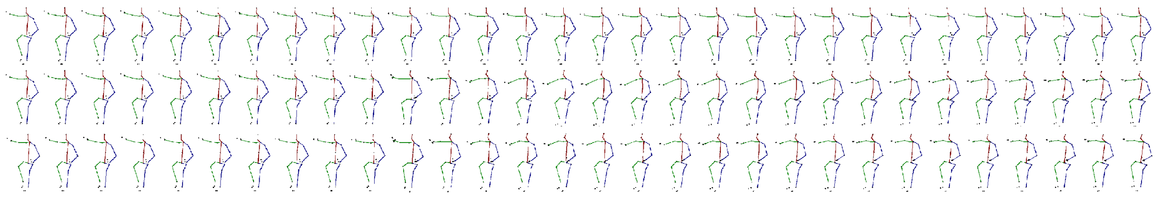
7.2 Results on prediction

Figure 7.1 shows some of the prediction results for both NTURGB-D and Human3.6M datasets. The top row is the ground truth, and each subsequent row corresponds to the predicted human poses from different z values, drawn from a uniform distribution. The input sequence of poses is of size 10 and the generator predicts 20 output poses for Figures 7.1a and 7.1b. The input is of size 10 and the prediction is of size 40 for Figure 7.1c.

These size parameters are specified to the training, and are configurable. Because we use a sequence-to-sequence recurrent network in the generator, the input and output sequence lengths can be different between the training and the testing phases. Each z value generates a separate possible future sequence of human poses.

We note that the first few predicted poses are always very close to the ground truth, which is to be expected. But as the model predicts more poses in the future, they start to diverge from the ground truth. Even though in our experiments we noted that some of the generated z values continued to produce predictions very close to the ground truth, we found no good way to control this, other than by using a brute force approach (i.e., generating many predictions and then picking the closest one to the ground truth).


(a) Sample prediction from NTURGB-D dataset, with 10 human poses as input and 20 human poses as output.


(b) Sample prediction from Human3.6M dataset, with 10 human poses as input and 20 human poses as output.

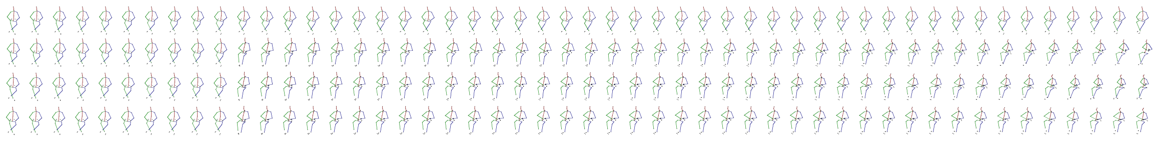
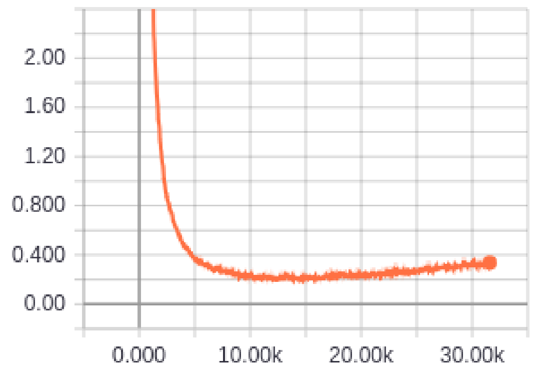

(c) Sample prediction from Human3.6M dataset, with 10 human poses as input and 40 human poses as output.

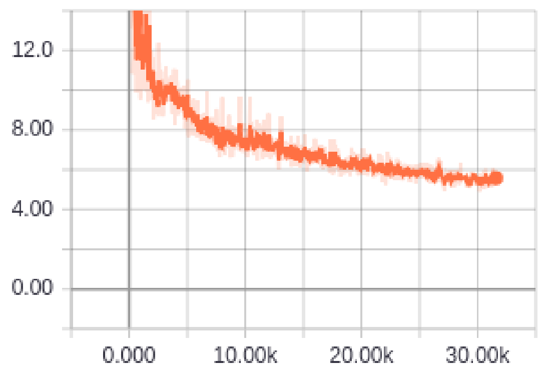
Figure 7.1: Example of pose prediction from the generator, input 10 poses and output 20 poses for the first two subfigures, and input 10 poses and output 40 poses for the last one. First row in each group is the ground trut,h and the other two rows are predicted poses, per z value.

In Figure 7.2, we plot the various losses as a function of the batch iteration. Compared to the work done by Barsoum, Kender, and Liu 2018 using WGAN-GP, our losses converge and the convergence values map to the quality of the prediction, due to the combination of simple GAN plus a gradient penalty (“GAN-GP”).

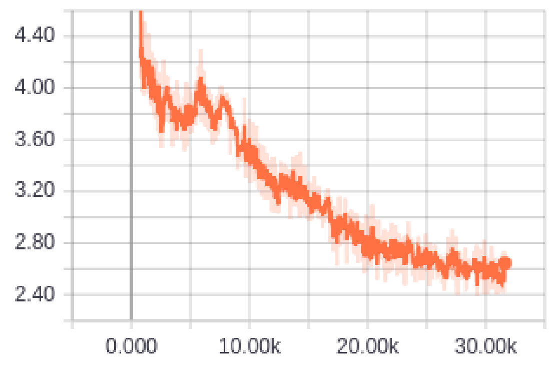
Figure 7.2 shows various losses for the training with the NTURGB-D entire



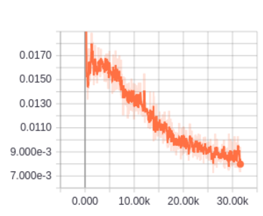
(a) Discriminator loss



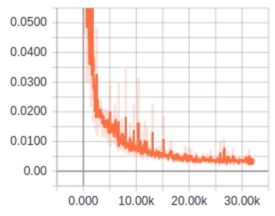
(b) Generator loss



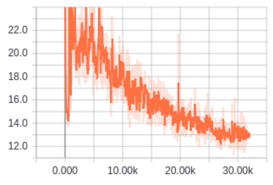
(c) Quality loss



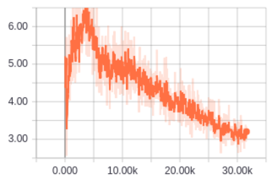
(d) Consistency



(e) Energy



(f) Bone



(g) GAN

Figure 7.2: Plots of various losses during training for the NTURGB-D dataset. The x axis corresponds to the step.

dataset. The generator loss in Figure 7.2b is the sum of multiple losses, four of them shown in Figures 7.2e, 7.2d, 7.2f, 7.2g.

In actuality, we found that the bone loss should not be used in NTURGB-D training; we have simply plotted the computed bone loss values here. Bone loss was better used in the Human3.6m (MoCap) dataset, instead of in the NTURGB-D (Kinect) dataset–where joint positions varied from frame to frame, based on inaccuracies in the Kinect's skeleton tracking pipeline. Some of these ground truth poses were very bad, as shown in Figure 7.3, especially when there was occlusion or little motion, or when there was a sideways pose obscuring an arm or a leg. But even without explicitly incorporating the bone loss in the generator loss function, the model here has learned accurate bone lengths, even when it is just computed directly between each joint in the generated poses.

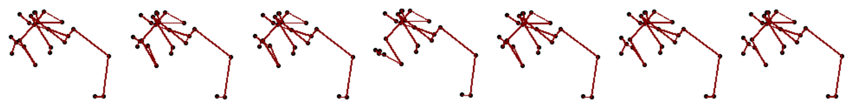


Figure 7.3: Bad skeleton data from NTURGB-D.

We note that the quality loss also converged. It therefore can be used to measure the quality of the prediction, as shown with the four sequences in Figure 7.4. As the visual quality of the prediction worsens, the computed probability of a valid human pose also approaches zero. The accuracy of this estimation enabled us to select the best model from the training.

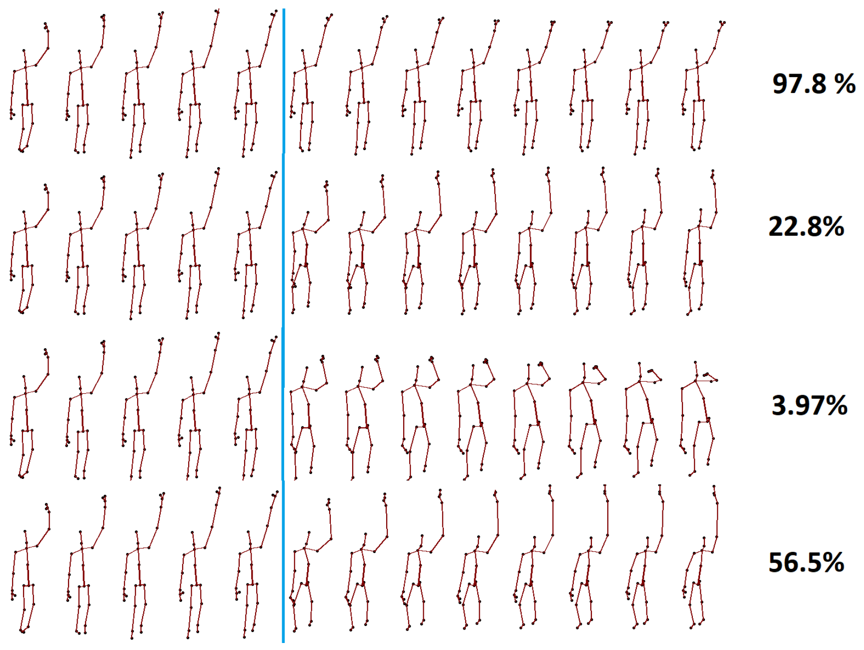


Figure 7.4: The probability that a sequence generated from the first five poses is a valid human motion.

As an example, the third row in the figure, with low quality probability 3.97%, generates bone lengths that are visually different than the ground truth, and the motion looks unnatural. We can easily see that the other rows provide better human poses, and that the higher the value, the better the quality of the poses across the time axis. For example, one difference between the first and second rows is that the transition from the ground truth in the first row is better, as shown in the poses before and after the blue line.

More detailed assessments of the quality network are shown in Table 7.1, which shows the probability output of the quality network for the ground truth pose and 10 different predictions (under a different z value per column), at various epochs (per row).

Epoch	Real	Human pose predictions for different z values									
		1	2	3	4	5	6	7	8	9	10
1	0.48	0.51	0.51	0.51	0.51	0.51	0.51	0.51	0.51	0.51	0.51
2	0.71	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57
13	0.96	0.06	0.06	0.07	0.05	0.06	0.06	0.06	0.06	0.06	0.06
14	0.95	0.07	0.06	0.08	0.05	0.07	0.06	0.06	0.06	0.07	0.07
152	0.88	0.59	0.31	0.23	0.65	0.21	0.51	0.18	0.41	0.34	0.18

Table 7.1: The probability output of the quality assessment network after each epoch (per row) for the ground truth (Real) and 10 predictions (per column).

At Epoch 1, the quality network can not differentiate between real and non-real human poses, so the output is like flipping a coin with probability $p_i \approx 50\%$. But neither can the generator generate valid sequences of poses. At Epoch 13 and 14, the quality network quickly learns to discard the generated poses, and so we have the probability of the ground truth is high $\approx 95\%$ and the probability of the prediction is close to zero. At Epoch 152, the generator has learned to generate non-real human poses that are close to the real ones, so we have the probability of the predicted values is increasing, and some of the predicted poses exceed 50%.

7.2.1 Experiment list

Table 7.2 shows the list of experiments that we tried on both NTURGB-D and Human3.6m:

Experiment	Description
All classes	We trained on the entire dataset, on all classes and all subjects. We split train versus test set at random. This experiment was the slowest (2 weeks training time on Titan X GPU), but with the best result. And due to the amount of the data, some of the custom loss did not affect the result.
One class	On the other extreme, we also ran experiments for each class separately. This was the fastest, which allowed us to try multiple experiments quickly. The quality was reasonable but not as good as training on the entire dataset. For this case, the custom loss became more important for the prediction quality.
Many classes	In this middle ground, we used a subset of the classes in order to train faster. The results were of better quality compared to those on a single class.
Different camera viewpoints	We compared training from a single camera versus multiple cameras. As expected, training from single camera provided a more consistent result.
Subject-based versus activity-based	In these cases we trained on a subset of actors or activities, and tested on the remaining actors or activities. We found that splitting based on subject did not affect quality much, but splitting based on activities did, especially if some of the activities in the test set were very different than in the training set (i.e., sitting versus standing).

Table 7.2: List of experiments we tried.

Beside the experiments shown in Table 7.2, we also tried different splits between training and testing sets based on activities, on subjects, and at random. We evaluated the results based on the quality assessment network. One of our biggest challenges was that there was no easy way to quantify the quality of the result in a non-subjective manner.

We also experimented with various GAN algorithms, as shown in Table 7.3.

Algorithm	Description
GAN	Collapses very easy, but before collapsing the loss starts to converge.
GAN with Softplus	Same as GAN.
LSGAN (Least Squares GAN)	More stable than GAN, but does not produce good results.
LS-GAN (Loss-Sensitive GAN)	Same as LSGAN.
WGAN (Wasserstein GAN)	Does not collapse, but produces bad results due to clipping of weights.
WGAN-GP	Does not collapse and produces good results, but predictions need to be manually inspected (generator loss value does not reflect quality).

Table 7.3: List of GAN algorithms that we tried.

We found that most stability issues were addressed by augmenting GAN algorithms shown in Table 7.3 with gradient penalty from WGAN-GP. Our latest network uses the original GAN with gradient penalty:"GAN-GP".

7.2.2 Failure cases

There are some cases that the model did not have enough information to generalize well. For example, when splitting based on activity, if the activity in the test set is very different from the one in the training set, causes inaccuracy in the prediction (e.g., if the training set does not have sitting poses, but the test set haves them). Another challenge is predicting more than 60 frames: the model prediction starts to fall apart. One workaround for this scenario is to predict a shorter sequence and then feed it back as input. A further problem is that the training stability is sensitive to the complexity of the model architecture and to the initialized weights.

7.3 Results on classification

We verify that the features learned by the discriminator are generic human motion features and can be applied to other related tasks. To do so, we train a classifier using the discriminator network with the last two layers replaced as shown previously in Figure 6.5. We then train this classifier twice, once using the parameters learned from the unsupervised training, and another with the parameters initialized at random (i.e. training from scratch).

7.3.1 NTURGB-D and Human3.6m

As shown in Figure 7.5, we first trained on the 49 classes of the NTURGB-D dataset. The blue plot, which uses the discriminator weights, starts at around 27.5% accuracy and reaches a steady state at around epoch 20. The red plot, which was trained from scratch, starts at much lower accuracy and takes about twice the number of epochs to reach a steady state.

Figure 7.6 and Figure 7.7 show the confusion matrix of the classifier at epoch 10 and epoch 50, respectively. As shown in both figures, the diagonal of the confusion matrix is brighter, and there are fewer off-diagonal values, when starting with the pre-trained discriminator. In fact, if we look at the Figure 7.6 right confusion matrix, label 9 has no valid prediction yet.

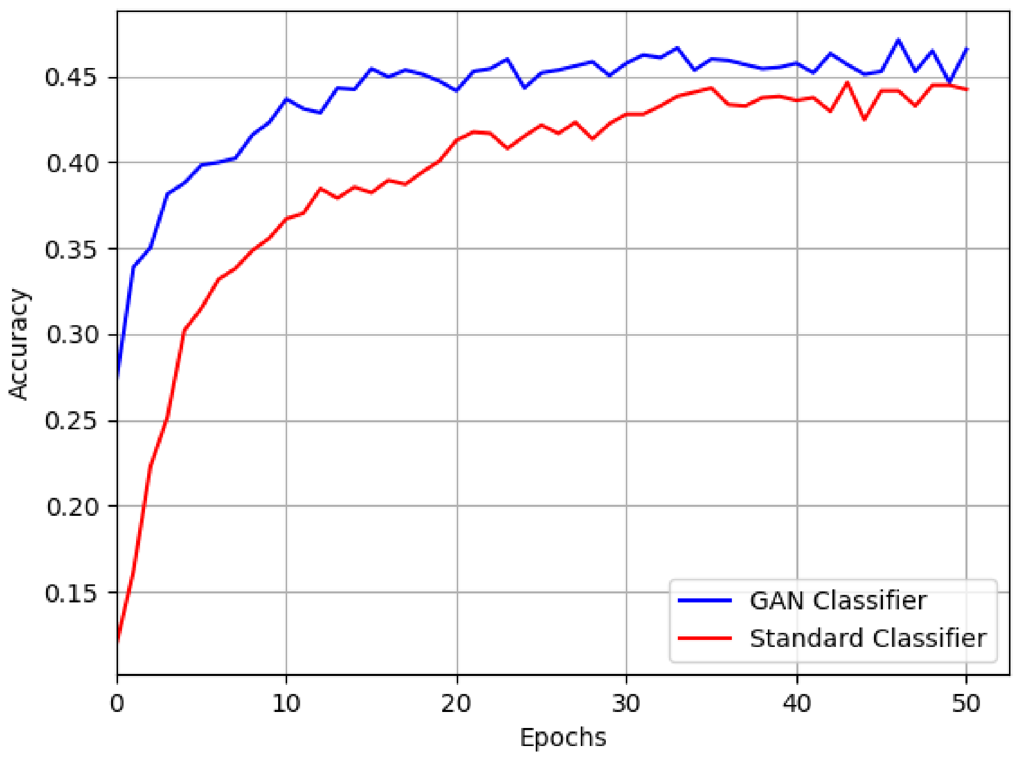


Figure 7.5: NTURGB-D classifier. The blue plot is the classifier initialized with the discriminator weights, and the red plot is the classifier initialized at random. As shown, the one using the discriminator parameters starts better and reaches steady state much faster.

We did the same experiment on the Human3.6m dataset, as shown in Figure 7.8. Classification is more difficult in the Human3.6m dataset, since each clip is around four minutes long. We used 30 frames to classify the entire clip, which not quite enough. The 30 frames are selected at random during training, and in the middle of the clip during testing, and they cover only 2 seconds duration. But even with that, we had better results with the pre-trained discriminator model than with training from scratch.

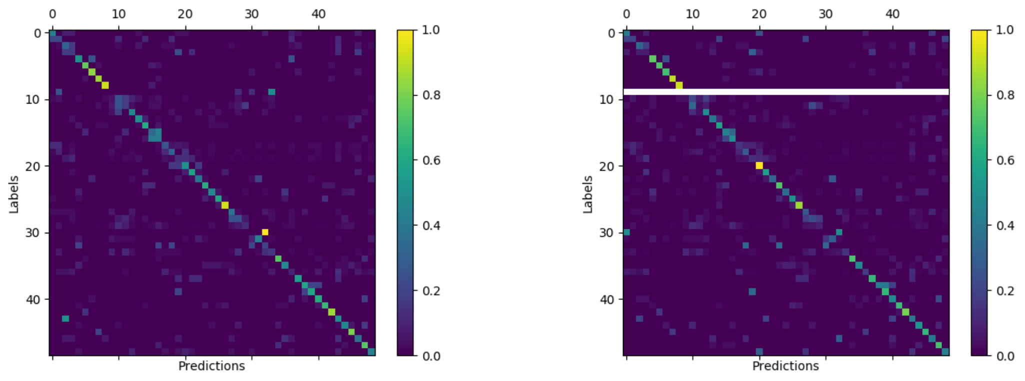


Figure 7.6: NTURGB-D classifier. Comparison between the confusion matrix at epoch 10. Left side is using the pre-trained discriminator and right side is training from scratch.

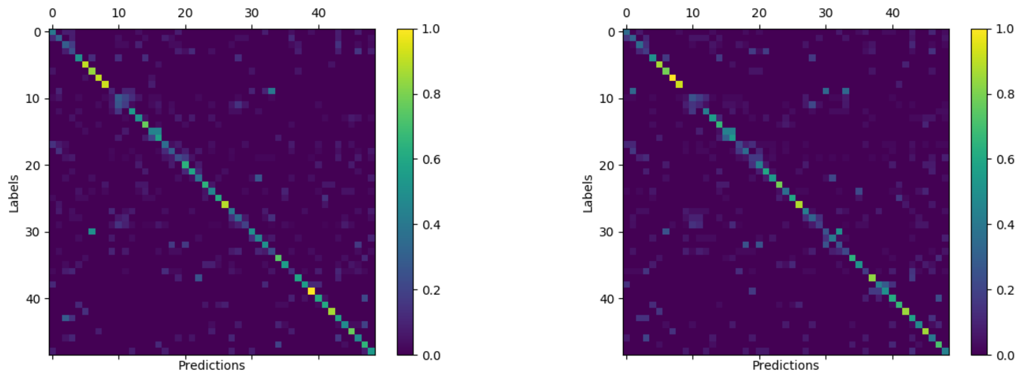
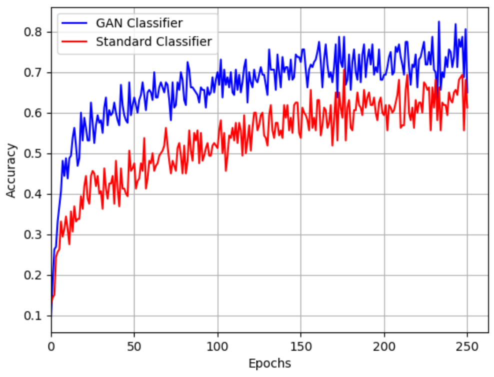


Figure 7.7: NTURGB-D classifier. Comparison between the confusion matrix at epoch 50. Left side is using the pre-trained discriminator and right side is training from scratch.

7.3.2 Unseen classes

Finally, to verify that the learned features are generic motion figures, valid even for action categories not seen during training, we removed the last 10 classes from the NTURGB-D dataset during GAN training, and restored them only in the classification training. As shown in Figure 7.9, the left confusion matrix was trained with the discriminator network and the right was trained from scratch. The orange rectangles



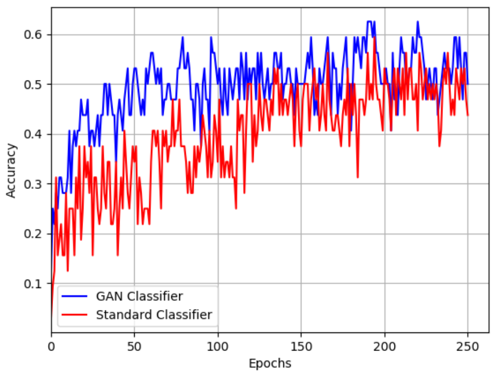
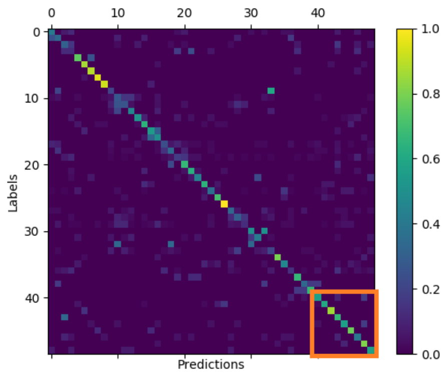


Figure 7.8: Human3.6m classifier. Accuracy plot, showing that starting with the pre-trained discriminator is better. Left plot is training data, and right plot is test data.

outline classes 40 to 49, which were the labels not used during the GAN training. Even within these rectangles, it is easy to see that the left confusion matrix has better accuracy and fewer off-diagonal terms than the corresponding area in the right confusion matrix.



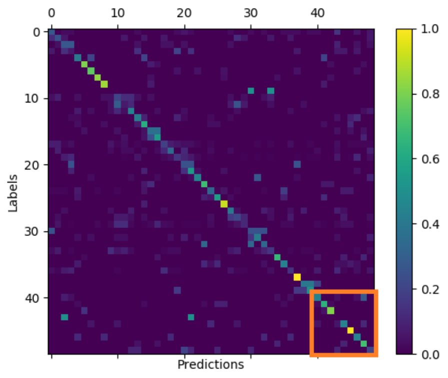


Figure 7.9: Human3.6m classifier. Comparison between the confusion matrix at epoch 12, with some classes not used in the GAN training. Left side is using the pre-trained discriminator and right side is training from scratch.

The maximum accuracy on the test data using the pre-trained weight of the discriminator is 49.20% at epoch 276, and the maximum accuracy training from scratch using the same network architecture is 48.64% at epoch 339.

7.3.3 Effect of reducing training data size

One of the advantages of using a pre-trained model on a different task is that the amount of labels needed for that task are smaller than training from scratch. We tested how good were the features learned by the discriminator as a result of GAN training on a reduced training data size. We reduced the training data by 50% and by 25%, then reran our experiments.

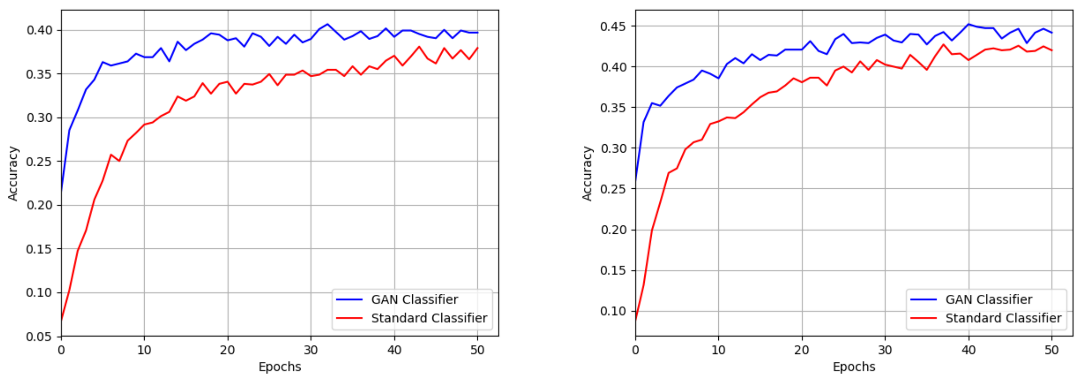
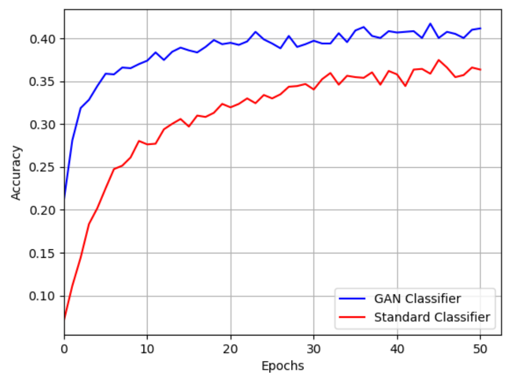


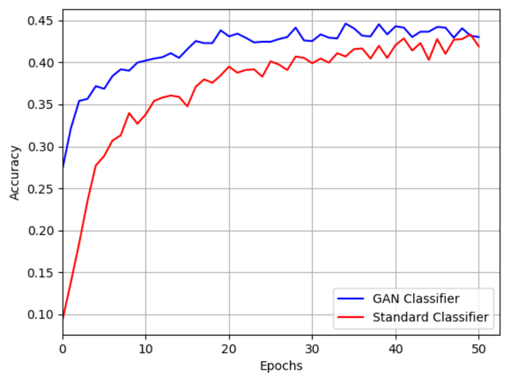
Figure 7.10: NTURGB-D accuracy with reduced training data. Left side, we reduced the training data by 50%, and right side, we reduced the training data by 25%, in order to show the effect of the pre-training on reduced labeled data.

As shown in Figure 7.10, when we reduced the training size by 50%, both training from the discriminator and training from scratch became less accurate. But the discriminator network still performed better, and reached 40% accuracy. Reducing the training data by 25%, the discriminator reached 46.71% accuracy, which is not far from training on the full dataset. Training from scratch reached only 42.07%, showing that the amount of training data needed using the pre-trained discriminator is less than training from scratch, while reaching the same accuracy.

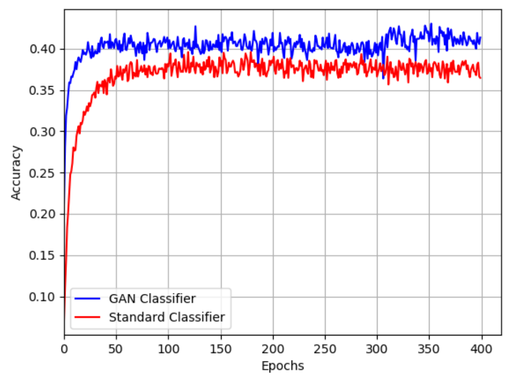
In the next experiment, we reduced the training data used by the classifier, but kept the full data for GAN training. The idea here is that we trained the GAN on a larger unsupervised dataset, but fine-tuned the trained model on a smaller labeled dataset.



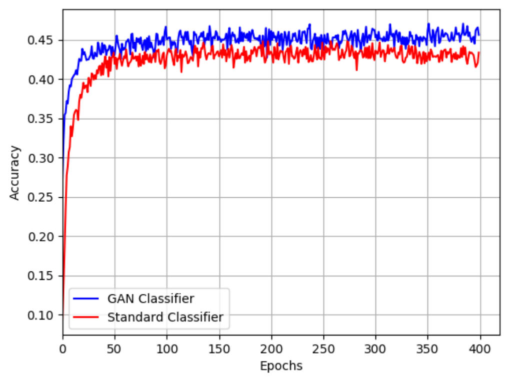
(a) Accuracy after 50 epochs with 50% reduction in training data.



(b) Accuracy after 50 epochs with 25% reduction in training data.



(c) Accuracy after 400 epochs with 50% reduction in training data.



(d) Accuracy after 400 epochs with 25% reduction in training data.

Figure 7.11: NTURGB-D accuracy with reduced training data for the classification task only. Left side, we reduced the training data by 50%, and right side, we reduced the training data by 25%, in order to show the effect of the pre-training on reduced labeled data. Blue plot is the discriminator accuracy during fine-tuning, and red plot is training from scratch.

As shown in figure 7.11, we reduced the training data for the classification training, but trained on the full data using GAN. For 50% reduction, the discriminator reaches 43.03%, which is much higher than training from scratch, which reaches only 39.58%. when removing 25% of the training data, the discriminator reaches 47.04%, and training from scratch reaches only 44.79%.

7.4 Loss ablation studies

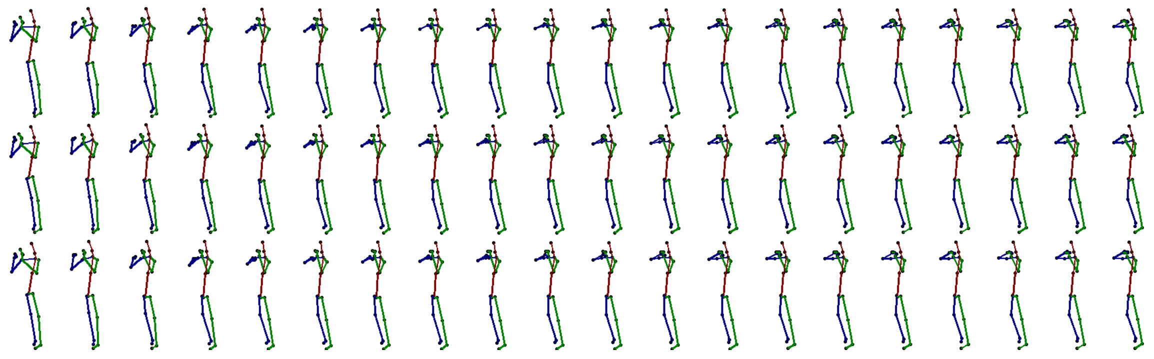
In this section, we discuss the results of our ablation studies, which investigated the effect of the individual terms used in our losses.

7.4.1 Gradient penalty

In GAN training without the gradient penalty, we could not get the model to converge, and it collapsed easily. However, when we tried other types of GAN algorithms (those that were not based on earth mover distance), all of them performed better with the gradient penalty. What the gradient penalty does is to lower the value of the norm of the gradient to 1.0, which enforces the Lipschitz constraint for the WGAN case. However, it showed better stability even on a standard GAN also, which does not need to adhere to the Lipschitz constraint. Our hypothesis is that the stability is the result of keeping the gradient value small. This area requires more investigation.

7.4.2 Divergence loss

Divergence loss controls the diversity of the predicted poses generated from the same input sequence, but with different z values. As shown in Figure 7.12, removing the divergence loss caused each z value to produce almost the same sequence of human poses. Increasing the coefficient of the divergence loss caused each z value to produce very different output sequences.


(a) Poses generated without divergence loss.

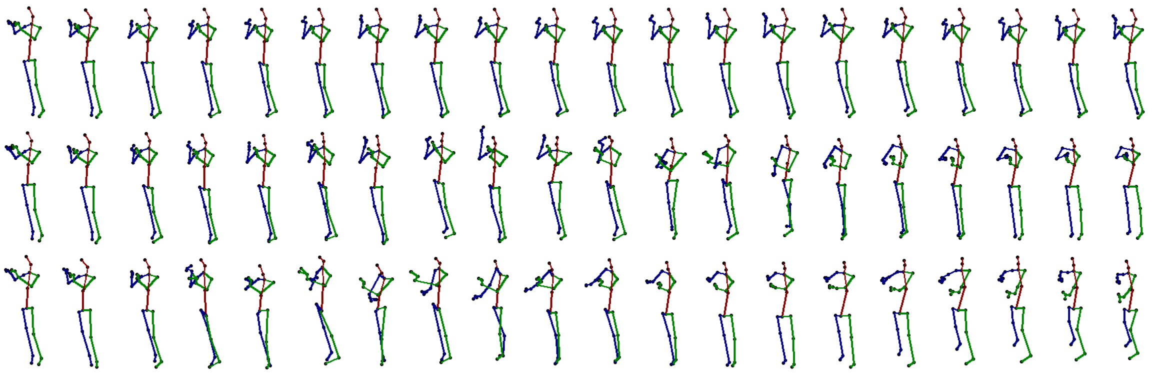
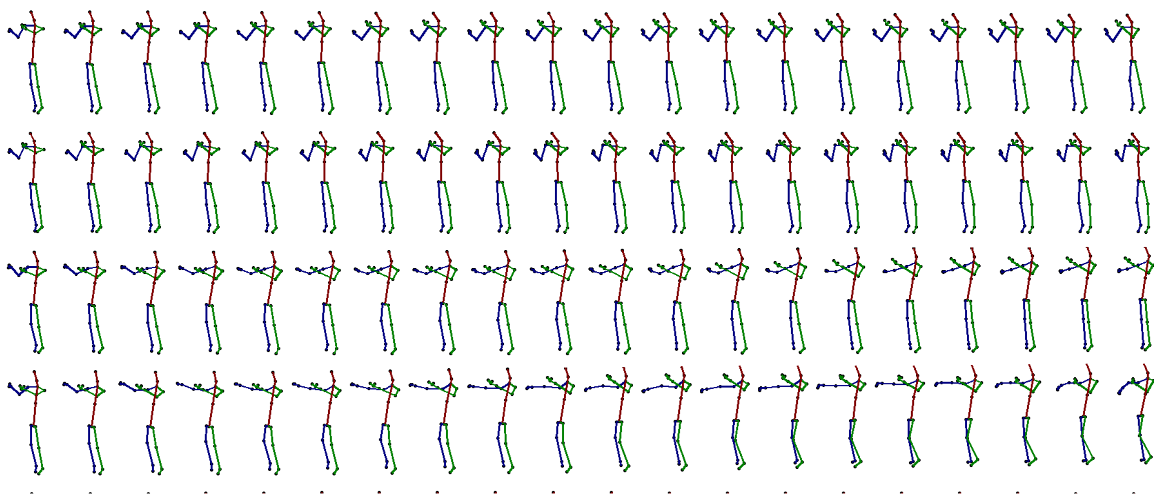

(b) Poses generated with divergence loss.

Figure 7.12: The effect of the divergence loss on the prediction result. Each row is from a different z value.

7.4.3 Pose gradient loss

This loss controls the pace of the motion between frames. Without it, we noticed jumpiness in the predicted poses in some cases.

We needed to be very careful while using consistency loss, because it can cause the model to simply copy the previous pose, such as $p(t) = p(t-1), \forall t$, if the weighting


(a) Poses generated without pose gradient loss.

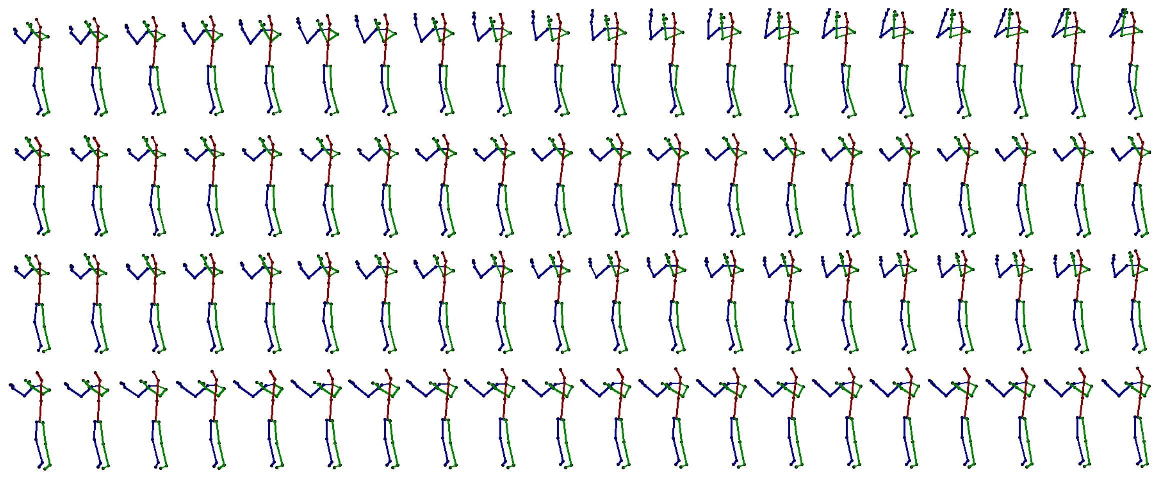

(b) Poses generated with pose gradient loss.

Figure 7.13: The effect of pose gradient loss on the prediction result. Each row is from a different z value.

factor α_{pg} is too high. This is the reason that we limit this loss value to not go beyond a certain threshold.

7.4.4 Energy loss

This loss computes and minimizes the energy expenditure in human motion, as used in animation to make motion smoother and more realistic Nathan et al. 2015. Based on that, we used energy loss to improve the quality of the predictor. However, looking at the energy equation, it minimizes nearly the same term as the pose gradient loss, except for the inclusion of an acceleration term. In multiple experiments, we noticed that the effects due to those two losses were very similar.

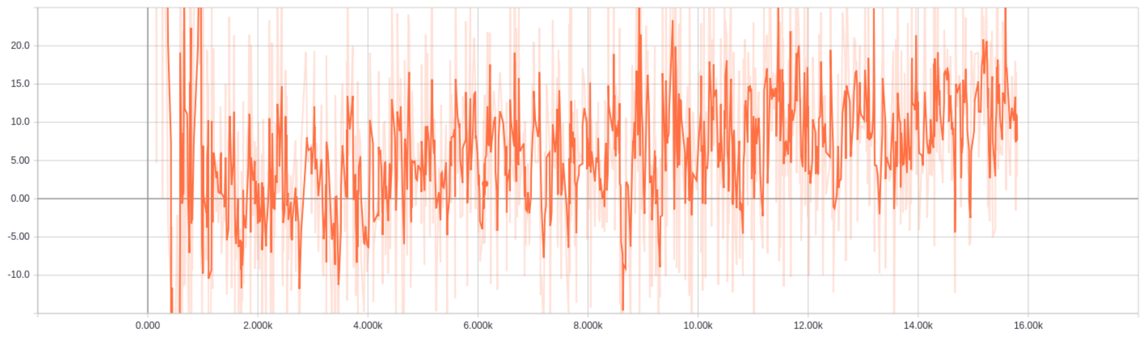
7.4.5 Bone loss

Bone loss affected the quality of the prediction negatively on NTURGB-D, but did not strongly affect the quality on Human3.6m. This is because there is some variation in bone lengths between each frame, which is more apparent in the NTURGB-D dataset due to the Kinect skeleton tracking algorithm. So in most of our training, we did not use bone loss.

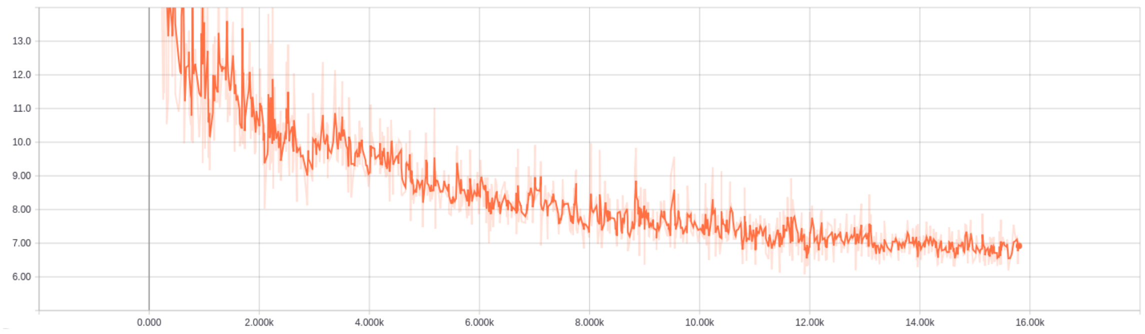
7.5 Comparing GAN versus WGAN-GP

The gradient penalty was introduced in the improved WGAN version, WGAN-GP, in order to satisfy a k-Lipschitz constraint on the gradient. In our experiments from Barsoum, Kender, and Liu 2018, WGAN-GP was in fact more stable than the original GAN, and it did not collapse during training, which was a recurrent problem with a pure GAN. However, when we added the same gradient penalty to the original GAN, it become more stable also. With that update, both WGAN-GP and the augmented GAN ("GAN-GP") produced similar human pose quality.

But still, based on multiple experiments on both the NTURGB-D and Human3.6M datasets, we found that the loss of WGAN-GP did not indicate the quality of the generator. And, even after reaching a good prediction quality, it diverged if we continued training. We did not have this problem with the augmented GAN, as shown in Figure 7.14.



(a) Generator loss using WGAN-GP approach.



(b) Generator loss using GAN approach with gradient penalty.

Figure 7.14: A comparison of the generator loss between WGAN-GP, and GAN with gradient penalty ("GAN-GP"). As shown, in the WGAN-GP case we cannot tell if the model converges or not. But in the augmented GAN case, we can measure the convergence of the model without resorting to manually inspecting the predictions.

The WGAN-GP loss is shown in Figure 7.14a. The loss moderated somewhat at the beginning, but then kept changing on each subsequent iteration. In contrast, the loss using the augmented GAN is shown in Figure 7.14b. We can easily see that the loss converges and stabilizes after about iteration $14K$. And, when we continued training further, the loss stayed convergent.

Conclusion

8.1 Summary

In this thesis, we focused on the problems of human pose anticipation and understanding. We first explored existing techniques in video recognition and unsupervised learning. We also discussed in depth how to improve data labeling on subjective datasets, such as those used in emotion recognition from facial appearance. We demonstrated that improved labels improved the quality of the model. We showed multiple ways to train datasets with multiple labels per image. We measured the benefit of using all the labeling information, instead of the common method of selecting only the top label using $argmax$.

For human motion anticipation and prediction, we have shown a novel sequence-to-sequence model for human motion prediction, with the ability to control the prediction behavior. We demonstrated how it can predict multiple plausible future of human poses from the same input. We also showed the representation power learned by the discriminator, which can be used in other related tasks such as action classification. To quantify the quality of the non-deterministic predictions, we simultaneously trained a motion quality assessment model, which learned the probability that a given skeleton sequence is a real human motion. We tested our architecture on two different datasets, one based on the Kinect sensor, and the other based on MoCap data. Experiments showed that our model performs well on both datasets.

8.2 Future work

Building upon this work, there are multiple streams of research that we are interested in pursuing.

8.2.1 The meaning of z

We plan to investigate the semantic meaning and state space of the z vector. If we can compute the reverse mapping from pose sequence to z, we might be able to use z values directly for action classification or clustering. One idea to compute the reverse mapping, similar to CycleGAN Zhu et al. 2017, is to have another network that computes the reverse. It would have two losses, one for z-to-poses and another for poses-to-z, to make sure the generated z or poses match the non-generated ones.

8.2.2 Data augmentation

Not every dataset has enough labels for every class. We would explore how to generate additional human poses for classes with low amounts of data. This would help balance training, while avoiding existing techniques which simply add more weights to classes that have a low number of samples. This would also improve the diversity of human poses for each category.

8.2.3 Reinforcement learning

Reinforcement learning is one of the areas in which future prediction will help the most. Since our architectures are able to anticipate the future, a system using them can compute future rewards from that future, reducing the amount of actual data needed to train a reinforcement learning algorithm.

8.2.4 Computing Confidences

Because we can generate multiple possible futures from the same input, we can also compute the confidence or probability of a certain action. For example, in monitoring a crowded scene for anomalies, we can generate n possible predictions from the current input, and then classify each prediction. From this, we can compute the probability of outcomes per class. As long as n is large, we can use this probability as the confidence of each action category, and detect unusual actions.

8.2.5 End-to-end training

One of the limitations that we found was nature of the datasets. NTURGB-D and Human3.6m are two of the largest datasets from a $3D$ source for action recognition. Still, all their actions are indoors, in the middle of the screen, with a static background, performed by non-realistic acting.

There has been much progress is generating $2D$ and $3D$ skeleton from standard $2D$ RGB videos Martinez et al. 2017; Newell, Yang, and Deng 2016. That work would enable us to take advantage of the large existing corpora of recordings in the wild, which is very difficult to achieve with standard limited depth sensors. Instead of using skeletons as an input, we could use the raw video frames directly and train end-to-end, bypassing the problems of depending on another skeleton pipeline.

8.2.6 Synthetic data

The quality of synthetic data has been improving dramatically in the last couple of years, due to progress in graphic algorithms and to improvements in Graphics Processing Unit (GPU) hardware. The advantages of using synthetic data for training is tremendous for human pose estimations. With accurate simulators, we can accurately label human poses under different circumstances and with occlusion. This offers great flexibility, and already we see papers (Sankaranarayanan et al. 2018, Atapour-Abarghouei and Breckon 2018, Tremblay et al. 2018) working on synthetic data for various vision tasks.

Bibliography

Arjovsky, Martín, Soumith Chintala, and Léon Bottou (2017). "Wasserstein GAN." In: *CoRR* abs/1701.07875. URL: `http://arxiv.org/abs/1701.07875`.

Atapour-Abarghouei, Amir and Toby P. Breckon (2018). "Real-Time Monocular Depth Estimation Using Synthetic Data With Domain Adaptation via Image Style Transfer." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Baccouche, Moez et al. (2011). "Sequential Deep Learning for Human Action Recognition." In: *Human Behavior Unterstanding - Second International Workshop, HBU 2011, Amsterdam, The Netherlands, November 16, 2011. Proceedings*, pp. 29–39. URL: `https://doi.org/10.1007/978-3-642-25446-8_4`.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). "Neural Machine Translation by Jointly Learning to Align and Translate." In: *CoRR* abs/1409.0473. URL: `http://arxiv.org/abs/1409.0473`.

Ballas, Nicolas et al. (2015). "Delving Deeper into Convolutional Networks for Learning Video Representations." In: *CoRR* abs/1511.06432. URL: `http://arxiv.org/abs/1511.06432`.

Bargal, Sarah et al. (2016). "Emotion Recognition in the Wild from Videos using Images." In: *International Conference on Multimodal Interaction (ICMI).*

Barsoum, Emad, John Kender, and Zicheng Liu (2018). "HP-GAN: Probabilistic 3D Human Motion Prediction via GAN." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops.*

Barsoum, Emad et al. (2016). "Training deep networks for facial expression recognition with crowd-sourced label distribution." In: *Proceedings of the 18th ACM International Conference on Multimodal Interaction.* ACM, pp. 279–283.

Bütepage, Judith et al. (2017). “Deep representation learning for human motion prediction and classification.” In: *CoRR* abs/1702.07486. arXiv: `1702.07486`. URL: `http://arxiv.org/abs/1702.07486`.

Carpenter, Gail A. and Stephen Grossberg (1988). “The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network.” In: *Computer* 21.3, pp. 77–88. ISSN: 0018-9162. URL: `http://dx.doi.org/10.1109/2.33`.

Chen, Baoyang et al. (2017). “Video Imagination from a Single Image with Transformation Generation.” In: *CoRR* abs/1706.04124. URL: `http://arxiv.org/abs/1706.04124`.

Chen, Ming yu and Alexander Hauptmann (2009). “Mosift: Recognizing human actions in surveillance videos.” In: *CMU-CS-09-161*.

Chung, Junyoung et al. (2014). “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.” In: *CoRR* abs/1412.3555. eprint: `1412.3555`. URL: `http://arxiv.org/abs/1412.3555`.

Donahue, Jeff et al. (2015). “Long-term recurrent convolutional networks for visual recognition and description.” In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 2625–2634. DOI: `10.1109/CVPR.2015.7298878`. URL: `https://doi.org/10.1109/CVPR.2015.7298878`.

Fakoor, Rasool et al. (2016). “Memory-augmented Attention Modelling for Videos.” In: *CoRR* abs/1611.02261. URL: `http://arxiv.org/abs/1611.02261`.

Fedus, William et al. (2017). “Many Paths to Equilibrium: GANs Do Not Need to Decrease aDivergence At Every Step.” In: *arXiv preprint arXiv:1710.08446*.

Fernando, Basura and Stephen Gould (2016). “Learning End-to-end Video Classification with Rank-Pooling.” In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pp. 1187–1196. URL: `http://jmlr.org/proceedings/papers/v48/fernando16.html`.

Fragkiadaki, Katerina et al. (2015). “Recurrent Network Models for Human Dynamics.” In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 4346–4354. URL: `https://doi.org/10.1109/ICCV.2015.494`.

Goodale, Melvyn A. and A.David Milner (1992). “Separate visual pathways for perception and action.” In: *Trends in Neurosciences* 15.1, pp. 20 –25. ISSN: 0166-

2236. DOI: https://doi.org/10.1016/0166-2236(92)90344-8. URL: http://www.sciencedirect.com/science/article/pii/0166223692903448.

Goodfellow, Ian J et al. (2013). "Challenges in representation learning: A report on three machine learning contests." In: *Neural information processing*. Springer, pp. 117–124.

Goodfellow, Ian et al. (2014). "Generative Adversarial Nets." In: *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Graves, Alex (2013). "Generating Sequences With Recurrent Neural Networks." In: *CoRR* abs/1308.0850. URL: http://arxiv.org/abs/1308.0850.

Guadarrama, S. et al. (2013). "YouTube2Text: Recognizing and Describing Arbitrary Activities Using Semantic Hierarchies and Zero-Shot Recognition." In: *2013 IEEE International Conference on Computer Vision*, pp. 2712–2719. DOI: 10.1109/ICCV.2013.337.

Gulrajani, Ishaan et al. (2017). "Improved Training of Wasserstein GANs." In: *CoRR* abs/1704.00028. URL: http://arxiv.org/abs/1704.00028.

Haque, Albert, Alexandre Alahi, and Li Fei-Fei (2016). "Recurrent Attention Models for Depth-Based Person Identification." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Harrison, J., R. A. Rensink, and M. van de Panne (2004). "Obscuring length changes during animated motion." In: *ACM Transactions on Graphics* 23.3, pp. 569–573.

He, Kaiming et al. (2015). "Deep Residual Learning for Image Recognition." In: *CoRR*.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory." In: *Neural Computation* 9.8, pp. 1735–1780. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

Hodgins, J. K., J. F. O'Brien, and J. Tumblin (1998). "Perception of human motion with different geometric models." In: *IEEE Transactions on Visualization and Computer Graphics* 4.4, pp. 307–316.

Ionescu, Catalin, Fuxin Li, and Cristian Sminchisescu (2011). "Latent Structured Models for Human Pose Estimation." In: *International Conference on Computer Vision*, pp. 2220–2227.

Ionescu, Catalin et al. (2014). "Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.7, pp. 1325–1339.

Jain, Ashesh et al. (2016). "Structural-RNN: Deep Learning on Spatio-Temporal Graphs." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 5308–5317. URL: https://doi.org/10.1109/CVPR.2016.573.

Ji, S. et al. (2013). "3D Convolutional Neural Networks for Human Action Recognition." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.1, pp. 221–231. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.59.

Jia, Xu et al. (2015). "Guiding the Long-Short Term Memory Model for Image Caption Generation." In: *ICCV*.

Karpathy, Andrej et al. (2014). "Large-scale Video Classification with Convolutional Neural Networks." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kim, T. K., S. F. Wong, and R. Cipolla (2007). "Tensor Canonical Correlation Analysis for Action Classification." In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. DOI: 10.1109/CVPR.2007.383137.

Kingma, Diederik P. and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization." In: *CoRR* abs/1412.6980. arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

Kiros, Ryan, Ruslan Salakhutdinov, and Richard S. Zemel (2014). "Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models." In: *CoRR* abs/1411.2539. URL: http://arxiv.org/abs/1411.2539.

Kohonen, Teuvo (1982). "Self-organized formation of topologically correct feature maps." In: *Biological Cybernetics* 43.1, pp. 59–69. ISSN: 1432-0770. URL: http://dx.doi.org/10.1007/BF00337288.

Koppula, Hema Swetha and Ashutosh Saxena (2013). "Learning Spatio-Temporal Structure from RGB-D Videos for Human Activity Detection and Anticipation." In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pp. 792–800. URL: http://jmlr.org/proceedings/papers/v28/koppula13.html.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks." In: *Advances in neural information processing systems*, pp. 1106–1114. URL: http://papers.nips.cc/

paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.

Lake, Brenden M. et al. (2017). "Building machines that learn and think like people." In: *Behavioral and Brain Sciences* 40, e253. DOI: 10.1017/S0140525X16001837.

Laptev, I. et al. (2008). "Learning realistic human actions from movies." In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. DOI: 10.1109/CVPR.2008.4587756.

Lehrmann, A. M., P. V. Gehler, and S. Nowozin (2014). "Efficient Nonlinear Markov Models for Human Motion." In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1314–1321. DOI: 10.1109/CVPR.2014.171.

Li, Chen et al. (2018). "Convolutional Sequence to Sequence Model for Human Dynamics." In: *CoRR* abs/1805.00655. eprint: 1805.00655. URL: http://arxiv.org/abs/1805.00655.

Liang, Xiaodan et al. (2017). "Dual Motion GAN for Future-Flow Embedded Video Prediction." In: *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1762–1770.

Liu, Jingen and M. Shah (2008). "Learning human actions via information maximization." In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. DOI: 10.1109/CVPR.2008.4587723.

Liu, Weitang, Emad Barsoum, and John D. Owens (2018). "Object Localization and Motion Transfer learning with Capsules." In: *CoRR* abs/1805.07706. arXiv: 1805.07706. URL: http://arxiv.org/abs/1805.07706.

Mao, Xudong et al. (2016). "Multi-class Generative Adversarial Networks with the L2 Loss Function." In: *CoRR* abs/1611.04076. URL: http://arxiv.org/abs/1611.04076.

Martinez, Julieta, Michael J. Black, and Javier Romero (2017). "On human motion prediction using recurrent neural networks." In: *CVPR*.

Martinez, Julieta et al. (2017). "A simple yet effective baseline for 3d human pose estimation." In: *CoRR* abs/1705.03098. arXiv: 1705.03098. URL: http://arxiv.org/abs/1705.03098.

Mathieu, Michaël, Camille Couprie, and Yann LeCun (2015). "Deep multi-scale video prediction beyond mean square error." In: *CoRR* abs/1511.05440. URL: http://arxiv.org/abs/1511.05440.

Mirsamadi, Seyedmahdad, Emad Barsoum, and Cha Zhang (2017). “Automatic Speech Emotion Recognition Using Recurrent Neural Networks with Local Attention.” In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Nathan, David et al. (2015). “Estimating Physical Activity Energy Expenditure with the Kinect Sensor in an Exergaming Environment.” In: *PLOS ONE* 10.5, pp. 1–22. DOI: `10.1371/journal.pone.0127113`. URL: `https://doi.org/10.1371/journal.pone.0127113`.

Newell, Alejandro, Kaiyu Yang, and Jia Deng (2016). “Stacked Hourglass Networks for Human Pose Estimation.” In: *CoRR* abs/1603.06937. arXiv: `1603.06937`. URL: `http://arxiv.org/abs/1603.06937`.

Neyshabur, Behnam, Srinadh Bhojanapalli, and Ayan Chakrabarti (2017). “Stabilizing GAN Training with Multiple Random Projections.” In: *CoRR* abs/1705.07831. URL: `http://arxiv.org/abs/1705.07831`.

Ng, Joe Yue-Hei et al. (2015a). “Beyond Short Snippets: Deep Networks for Video Classification.” In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ng, Joe Yue-Hei et al. (2015b). “Beyond Short Snippets: Deep Networks for Video Classification.” In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Pan, Yingwei et al. (2016). “Jointly Modeling Embedding and Translation to Bridge Video and Language.” In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 4594–4602. URL: `https://doi.org/10.1109/CVPR.2016.497`.

Pavlovic, Vladimir, James M. Rehg, and John MacCormick (2000). “Learning Switching Linear Models of Human Motion.” In: *Proceedings of the 13th International Conference on Neural Information Processing Systems*. NIPS’00. Denver, CO: MIT Press, pp. 942–948. URL: `http://dl.acm.org/citation.cfm?id=3008751.3008888`.

Qi, Guo-Jun (2017). “Loss-Sensitive Generative Adversarial Networks on Lipschitz Densities.” In: *CoRR* abs/1701.06264. URL: `http://arxiv.org/abs/1701.06264`.

Reitsma, P. S. A. and N. S. Pollard (2003). “Perceptual metrics for character animation: sensitivity to errors in ballistic motion.” In: *ACM Transactions on Graphics* 22.3, pp. 537–542.

Ren, Liu et al. (2005). “A Data-Driven Approach to Quantifying Natural Human Motion.” In: *ACM Transactions on Graphics* 24.3, pp. 1090–1097.

Ren, Shaoqing et al. (2015). “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” In: *CoRR* abs/1506.01497. URL: http://arxiv.org/abs/1506.01497.

Rohrbach, M. et al. (2013). “Translating Video Content to Natural Language Descriptions.” In: *2013 IEEE International Conference on Computer Vision*, pp. 433–440. DOI: 10.1109/ICCV.2013.61.

Rosenthal, Robert (2005). “Conducting judgment studies: Some methodological issues.” In: *The new handbook of methods in nonverbal behavior research*, pp. 199–234. DOI: 10.1093/acprof:oso/9780198529620.003.0005.

Sabour, Sara, Nicholas Frosst, and Geoffrey E Hinton (2017). “Dynamic Routing Between Capsules.” In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., pp. 3856–3866. URL: http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.pdf.

Sadanand, S. and J. J. Corso (2012). “Action bank: A high-level representation of activity in video.” In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1234–1241. DOI: 10.1109/CVPR.2012.6247806.

Sankaranarayanan, Swami et al. (2018). “Learning From Synthetic Data: Addressing Domain Shift for Semantic Segmentation.” In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shahroudy, Amir et al. (2016a). “NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis.” In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shahroudy, Amir et al. (2016b). “NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis.” In: *CoRR* abs/1604.02808. URL: http://arxiv.org/abs/1604.02808.

Sharma, Shikhar, Ryan Kiros, and Ruslan Salakhutdinov (2015). “Action Recognition using Visual Attention.” In: *CoRR* abs/1511.04119. URL: http://arxiv.org/abs/1511.04119.

Shou, Zheng, Dongang Wang, and Shih-Fu Chang (2016). “Action Temporal Localization in Untrimmed Videos via Multi-stage CNNs.” In: *CoRR* abs/1601.02129. URL: http://arxiv.org/abs/1601.02129.

Sidenbladh, Hedvig, Michael J. Black, and Leonid Sigal (2002). "Implicit Probabilistic Models of Human Motion for Synthesis and Tracking." In: *Proceedings of the 7th European Conference on Computer Vision-Part I*. ECCV '02. London, UK, UK: Springer-Verlag, pp. 784–800. ISBN: 3-540-43745-2. URL: `http://dl.acm.org/citation.cfm?id=645315.649172`.

Simonite, Tom. *The Missing Link of Artificial Intelligence*. `https://www.technologyreview.com/s/600819/the-missing-link-of-artificial-intelligence`. Accessed: 2018-11-18.

Simonyan, Karen and Andrew Zisserman (2014a). "Two-Stream Convolutional Networks for Action Recognition in Videos." In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al., pp. 568–576. URL: `http://papers.nips.cc/paper/5353-two-stream-convolutional-networks-for-action-recognition-in-videos.pdf`.

— (2014b). "Very Deep Convolutional Networks for Large-Scale Image Recognition." In: *CoRR* abs/1409.1556. URL: `http://arxiv.org/abs/1409.1556`.

Sion, Maurice (1958). "On general minimax theorems." In: *Pacific J. Math.* 8.1, pp. 171–176. URL: `https://projecteuclid.org:443/euclid.pjm/1103040253`.

Srivastava, Nitish, Elman Mansimov, and Ruslan Salakhutdinov (2015). "Unsupervised Learning of Video Representations using LSTMs." In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 843–852. URL: `http://jmlr.org/proceedings/papers/v37/srivastava15.html`.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). "Sequence to Sequence Learning with Neural Networks." In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 3104–3112. URL: `http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks`.

Szegedy, Christian et al. (2015). "Going Deeper With Convolutions." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 00, pp. 1–9. DOI: `10.1109/CVPR.2015.7298594`. URL: `doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298594`.

Tremblay, Jonathan et al. (2018). "Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization." In: *CoRR* abs/1804.06516. eprint: `1804.06516`. URL: `http://arxiv.org/abs/1804.06516`.

Uehara, Masatoshi et al. (2016). "Generative Adversarial Nets from a Density Ratio Estimation Perspective." In: *CoRR* abs/1610.02920.

Wang, Heng et al. (2011). "Action Recognition by Dense Trajectories." In: *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition.* CVPR '11. IEEE Computer Society, pp. 3169–3176. ISBN: 978-1-4577-0394-2. DOI: `10.1109/cvpr.2011.5995407`. URL: `http://dx.doi.org/10.1109/CVPR.2011.5995407`.

Wang, Jack M., David J. Fleet, and Aaron Hertzmann (2005). "Gaussian Process Dynamical Models." In: *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pp. 1441–1448. URL: `http://papers.nips.cc/paper/2783-gaussian-process-dynamical-models`.

— (2008). "Gaussian Process Dynamical Models for Human Motion." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 30.2, pp. 283–298. DOI: `10.1109/TPAMI.2007.1167`. URL: `https://doi.org/10.1109/TPAMI.2007.1167`.

Wang, Xin, Qiudi Chen, and Wanliang Wang (2014). "3D Human Motion Editing and Synthesis: A Survey." In: *Computational and Mathematical methods in medicine* 2014.

Xiao, Han, Kashif Rasul, and Roland Vollgraf (2017). "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms." In: *CoRR* abs/1708.07747. eprint: `1708.07747`. URL: `http://arxiv.org/abs/1708.07747`.

Xie, Lingxi et al. (2016). "DisturbLabel: Regularizing CNN on the Loss Layer." In: *CoRR* abs/1605.00055. URL: `http://arxiv.org/abs/1605.00055`.

Xu, Kelvin et al. (2015). "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention." In: *CoRR* abs/1502.03044. URL: `http://arxiv.org/abs/1502.03044`.

Yao, Li et al. (2015a). "Describing Videos by Exploiting Temporal Structure." In: *The IEEE International Conference on Computer Vision (ICCV).*

Yao, Li et al. (2015b). "Video Description Generation Incorporating Spatio-Temporal Features and a Soft-Attention Mechanism." In: *CoRR* abs/1502.08029. URL: `http://arxiv.org/abs/1502.08029`.

Yeung, Serena et al. (2016). "End-to-End Learning of Action Detection from Frame Glimpses in Videos." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 2678–2687. URL: `https://doi.org/10.1109/CVPR.2016.293`.

Youtube 8M (2016 (accessed December 17, 2016)). `https://research.google.com/youtube8m`.

Yu, Zhiding and Cha Zhang (2015). "Image Based Static Facial Expression Recognition with Multiple Deep Network Learning." In: *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ICMI '15. Seattle, Washington, USA: ACM, pp. 435–442. ISBN: 978-1-4503-3912-4. DOI: `10.1145/2818346.2830595`. URL: `http://doi.acm.org/10.1145/2818346.2830595`.

Zhao, Hang et al. (2017). "SLAC: A Sparsely Labeled Dataset for Action Classification and Localization." In: *arXiv preprint arXiv:1712.09374*.

Zhu, Jun-Yan et al. (2017). "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks." In: *Computer Vision (ICCV), 2017 IEEE International Conference on*.