The Pennsylvania State University

The Graduate School

College of Engineering

PREDICTION AND ESTIMATION OF HUMAN MOTION USING GENERATIVE-ADVERSARIAL NETWORKS

A Thesis in

Electrical Engineering

by

Amogh Subbakrishna Adishesha

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science

August 2018

The thesis of Amogh Subbakrishna Adishesha was reviewed and approved* by the following:

Robert Collins
Associate Professor of Computer Science Engineering
Thesis Advisor

David J Miller
Professor of Electrical Engineering

Mehdi Kiani
Assistant Professor of Electrical Engineering

Kultegin Aydin
Professor of Electrical Engineering and Department Head

*Signatures are on file in the Graduate School

ABSTRACT

Prediction of the human motion model has been an intrinsic part of several applications over diverse fields like gaming, augmented reality and cinematic graphics. The ability to estimate motion, ahead of time, helps robots predict human action and thus reduce time to react effectively. In real time applications such as pedestrian motion prediction, the availability of long motion sequences at test time is rare. In this work, we propose a new architecture to predictively model human motion partially from noise. We utilize the data synthesizing ability of Generative Adversarial Networks(GANs) to provide artificial motion frames that help in prediction of the motion sequence in an LSTM-RNN framework. The well proven Recurrent Neural Network is used as a discriminator in training a weaker LSTM generator that we later exploit in creating ground truth like data from randomly sampled frames with mean pose and added noise. Pivoting on the evaluation metrics used in latest works, we discuss the recent motion prediction techniques and compare the results. We also evaluate the training procedures, input requirements and complexity of the structures, thus illustrating the simplicity and accuracy of a GAN based input reduction model.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to acknowledge my advisor, Dr Robert Collins, for being my intellectual role-model. He directed my research, taught me to observe and analyze data, formulate the problem and search for good solutions. For his assistance, I am in his debt. I had the fortune of being taught by excellent professors and teachers at every stage of my academic life. To each and every one of them I am grateful. I would like to thank Dr David Miller and Dr Mehdi Kiani for serving in my committee and giving many helpful suggestions. My friends shared in my joys and celebrations, and pulled me through the bad times (philosophical and computational). I wouldn't have succeeded without them. Most importantly, I want to thank my parents who raised me, taught me the joys of learning and showed me the way. To them, I dedicate this thesis.

DEDICATION

To my parents.

Chapter 1

Introduction

Motion, impartial to the object under consideration, is a time dependent system that can be understood as a fundamental method of interaction of objects in the universe. Motion capture and prediction has been a vital and fascinating tool for long. The prediction of human action and the predictive analysis of motion helps us advance in the fields of computer interaction, pedestrian detection, activity recognition, robotics and image based pose estimation. Motion modeling, which used to be a complex task, is often partially represented as an observation based learning procedure. With advancement in the field of neural networks, we arrive at new algorithms and techniques to predict this model with very little or almost no prior observation of data. The latest addition to this list of regression based networks is the GAN. The Generative Adversarial Network as introduced by Goodfellow et al [11] is a combination of two neural networks that "fight out" a min-max game, at the end of which, the generator network produces original like data samples that deceive the discriminator network.

GANs have been used over time for various purposes and many such have been explained in detail in the literature survey. In the original paper, Goodfellow et al explain the ability of GANs to generate synthetic versions of numbers of the MNIST dataset. GANs use an adversarial loss function to simultaneously train the generator and the discriminator. The goal of the network is to sample data from a noise distribution and transform it to the probability distribution of the dataset to be learnt. They are generally unstable and severely susceptible to changes in the hyper-parameters. In previous works involving GANs, minibatch training has been used to avoid the convergence of the output to the mean of the distribution.

In a motion prediction setting, the location of each joint over time can be seen as a sequence which varies with a non-deterministic nature. For this application and many other time dependent signals, Recurrent Neural Networks (RNNs) have been proven to be good at end to end learning. It was only recently that the ability of RNNs to understand spatio-temporal structure was exploited. [2]. Their ability to store relevant pattern information in memory has expanded their usage from stock price prediction to video sequence analysis.

Introduction to Long Short-Term Memory (LSTM):

If an RNN were to learn the pattern of a simple sine wave, it would need to store the frequency or the wavelength of the sine wave in memory in order to predict future sequences of the same pattern. This necessity to have a separate memory unit in a recurrent neural network led to the rise of LSTMs. LSTMS are used as building units for layers of a Recurrent Neural Network. A RNN composed of LSTM units is often called an LSTM network. A common LSTM unit has a cell, as shown in Figure 1-1, an input gate, an output gate and a forget gate. The cell stores certain values over time sequences and is called the memory of the network. Each of the three gates can be thought of as a "conventional" artificial neuron, as in a multi-layer (or feedforward) neural network: that is, they compute an activation (using an activation function) of a weighted sum. Intuitively, they can be thought of as regulators of the flow of values that go through the connections of the LSTM [8].

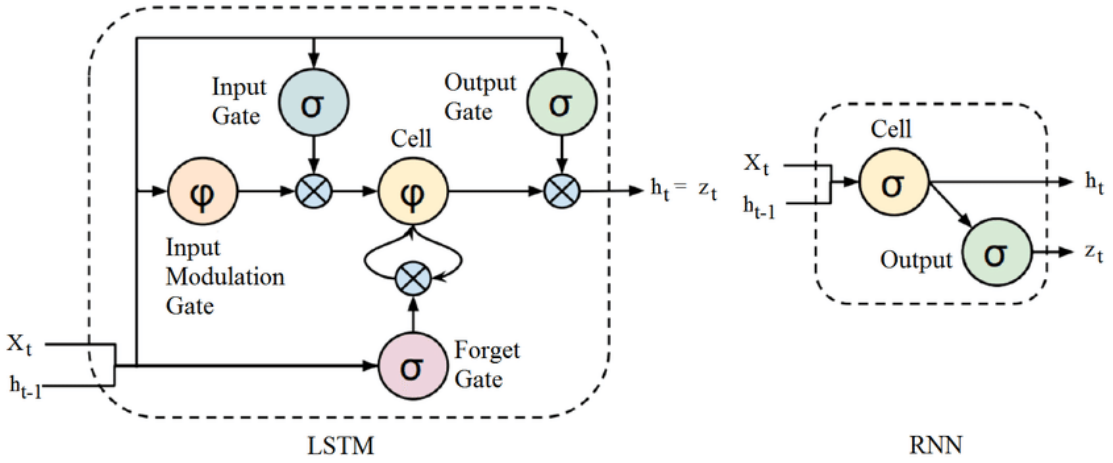


Figure **1-1:** LSTM cell with ht as ouput for input xt

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

In the above equations which mathematically describe the LSTM cell in Figure 1-1, f_t is the activation vector for the forget gate, i_t is activation vector for the input gate, o_t is the activation vector for the output gate, c_t is the current state of the cell and h_t is the output of the LSTM cell at time instance t.

There are multiple LSTM architectures in the literature. The most common one is the Vanilla LSTM which has a single layer and a simple memory unit. A CNN LSTM uses a Convolutional Neural Network to extract the features from an image or any data and that is fed over time to the Vanilla LSTM to get a sequence based output. Encoder-Decoder LSTM is one in which one LSTM encodes input sequences and a separate LSTM decodes output sequences. Most motion prediction algorithms use a sequence-sequence architecture (input is a time dependent sequence and output is a time dependent sequence) with separate encoder and decoder architecture where multiple LSTM cells are stacked together in layers to hold more data. RNNs are network models that process sequential data using

recurrent connections between their neural activations at consecutive time steps. There are two common methods used in the past with RNNs. The first being "Lagrangian" which predicts future outcomes conditioned on an object tracklet(a set of inputs) and the "Eulerian" which conditions on a tube (spatial region of interest) fixated at a particular pixel location. Lagrangian methods work well for motion prediction, sound/ stock prediction while Eulerian has proven to perform well for handwriting font learning where the spatial information under a sliding window helps in understanding a sequence as a whole.

Proposed network:

The proposed network has two major units. The data synthesis unit and the evaluation unit. In this work, the advancement is in the synthesis unit, which is articulated by reduction in input requirements over existing motion prediction techniques.

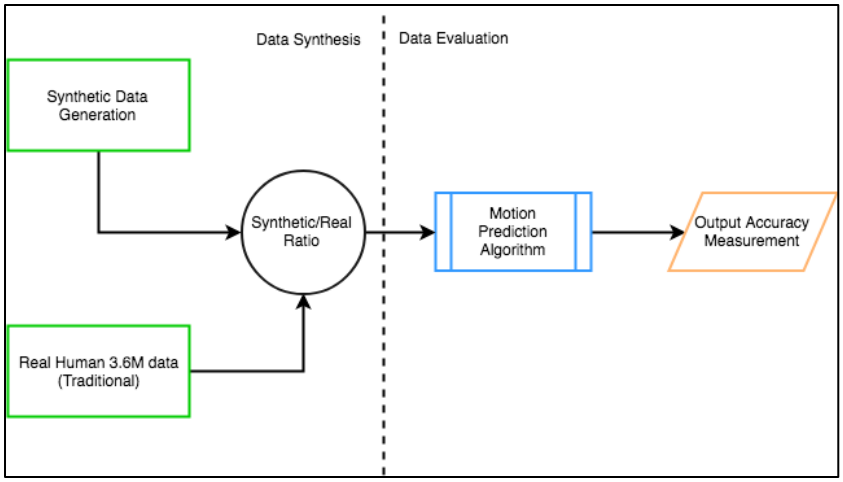


Figure 1-**2:** Proposed Technique

Figure 1-2 shows an abstraction of the procedure and the generalized evaluation technique. The primary goal is to reduce the input requirement of motion prediction algorithms while maintaining the accuracy that already exists. The secondary objective is to improve the accuracy of the motion prediction

system by using larger training data than previous systems. The trade-off point between data and accuracy is to be found empirically.

Chapter 2

Evaluation of Motion Prediction Models in Literature

The generation of naturalistic human motion sequences using probabilistic models trained on motion capture (MoCap) data has previously been addressed in the context of computer graphics, robotics, animation and machine learning. The several approaches to this problem divide largely into two parts: the probabilistic approach and LSTM-RNN approach. The probabilistic models often have complex training algorithms and cannot easily learn over different action sets. However, they can conveniently learn over probabilistic results. The probabilistic models include bilinear spatiotemporal basis models [22], linear dynamical systems [19], Hidden Markov Models [23], Gaussian process latent variable models [20], as well as multilinear variants thereof. Dynamical models based on Restricted Boltzmann Machines (RBMs) have been proposed for synthesis and infilling of motion data [21].

LSTM-RNN based models can easily be trained with a simple stochastic gradient descent method and can be evaluated very efficiently during test time with simple feedforward operations. They have been used for video pose labeling and forecasting of temporal context in kinematic tracking using dynamic programming over multiple per frame body pose hypotheses [19]. Vector based optical flow can only estimate the motion of body joints that do not move too fast and do not get occluded or dis-occluded. The time-based coupling in optical flow is pairwise, not long range. Encoder-Recurrent-Decoders (ERDs) aggregate information in time across multiple frames and use it to keep track of body parts as they become occluded and dis-occluded.

Parametric temporal filters such as Kalman filtering [16], Hidden Markov Models or Gaussian processes for action specific activities generally use simple, linear dynamics models for prediction. This includes the point acceleration, velocity and displacement

vectors of the body joints. These simple dynamics lose their accuracy over time and start to accumulate large errors, making it difficult to incorporate long range temporal information. Switching dynamic systems or HMMs detect activity transitioning and have been proven to have a greater capacity with increase of layers. However, their parameter count increases quadratically and this increases the computation time for this algorithm. In contrast, in ERD, action transitioning is transparent and also more effective. RNNs use distributed representations: each state can be seen as the ensemble of hidden activations in the recurrent layer. Thus, doubling the representation power (assuming binary units are used) by adding a single neural unit quadratically increases the number of parameters employed.

Deep RNNs for human motion

Multiple LSTM-RNN layers can be stacked to attain a better prediction of motion sequences. There is a trade-off between the number of layers of the RNN network and the computational efficiency the system has. Fragkiadaki et al used a 3 layered LSTM network (LSTM-3LR) which did not consider the hierarchical structure of the human body. This caused a drift to occur over time. They fixed this issue using an Encoder-Recurrent-Decoder network which had the data preprocessed in the form of Euler angles which were based on the hierarchical structure of the human body. In this network, they also introduced the technique of noise scheduling during training to avoid accumulation of errors. This causes the network to be more robust during testing. This noise scheduling, although hard in practice, makes the network able to generate realistic human motion for longer time horizons, especially on cyclic sequences. In the recent literature, Structural-RNN or SRNN [2] was used to fit a manually designed graph that encodes semantic knowledge about the RNN as input, and creates a bi-layer architecture that

assigns individual RNN units to semantically similar parts of the data. For example, the knee joints of both legs have similar RNN weight. This network also employs the noise scheduling technique introduced by Fragkiadaki et al. Their network outperforms previous work both quantitatively in short-term prediction, as well as qualitatively in terms of relative angle accuracy.

Taylor et al., proposed a conditional restricted Boltzmann machine (CRBM) that assumes a binary latent space and model motion using a CRBM which requires sampling for inference. Their non-linear generative model for human motion data uses an undirected model with binary latent variables and real-valued "visible" variables that represent joint angles. The latent and visible variables at each time step receive directed connections from the visible variables at the last few time-steps. This was among the first few approaches to perform spatio-temporal learning with human motion data. This particular technique is not used for comparison as the metrics of training, the absence of a RNN network and the use of raw input without normalization make it a standalone technique. Figure 2-1 shows the network as illustrated in their paper from 2006.

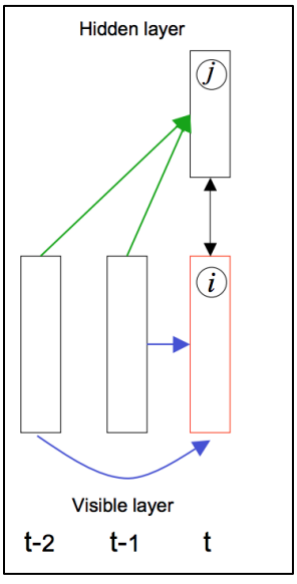


Figure **2 -1:** CRBM architecture for performing motion prediction

Encoder Recurrent Decoder (ERD)

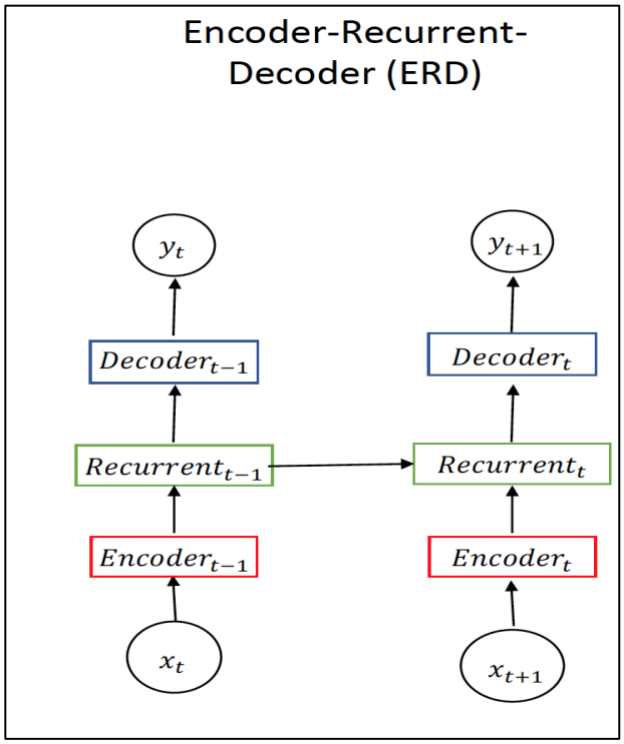


Figure **2-2**: ERD architecture as in Fragkiadaki et al (2015)

ERD is among the first RNN based approaches to motion prediction and was introduced by Fragkiadaki et al in 2015. Figure 2-2 illustrates ERD models for recurrent kinematic tracking and forecasting as shown in the original paper. They utilize the 3D body joint angles in a kinematic tree representation as a MoCap vector. They first represent each joint by an exponential map corresponding to 3 degrees of freedom per joint. Since knees do not move L-R, the respective rotation variables would have constants in them, reducing the data to be predicted and also incorporating the knowledge of the human body. The input vector of raw joint angles is standardized by first subtracting the mean and then dividing by the standard deviation along each dimension. The loss based approach for training is to reduce the Euclidian loss between the target and predicted joint angles.

Table 1: Error rates for ERD as reported in Fragkiadaki et al (2015)

LSTM-3LR	80ms	160ms	320ms	560ms	100ms
Walking	1.30	1.56	1.84	2.00	2.38
Eating	1.66	1.93	2.28	2.36	2.41
Smoking	2.34	2.74	3.73	3.68	3.82
Discussion	2.67	2.97	3.23	3.47	2.92

Table 1 reports the error rates, which are mean Euclidean distance between predicted and ground truth pairwise joints in frames averaged per frame and then averaged over 8 seed motion sequences over multiple epochs to maintain consistency. The error at 0ms is 0 at which point the ground truth input is cut off and the network starts predicting frames. These frames are fed back to the network to keep the prediction cycle going forward in time. This causes a buildup in error as every new predicted frame with error is assumed to be clean input for the prediction of the next frame.

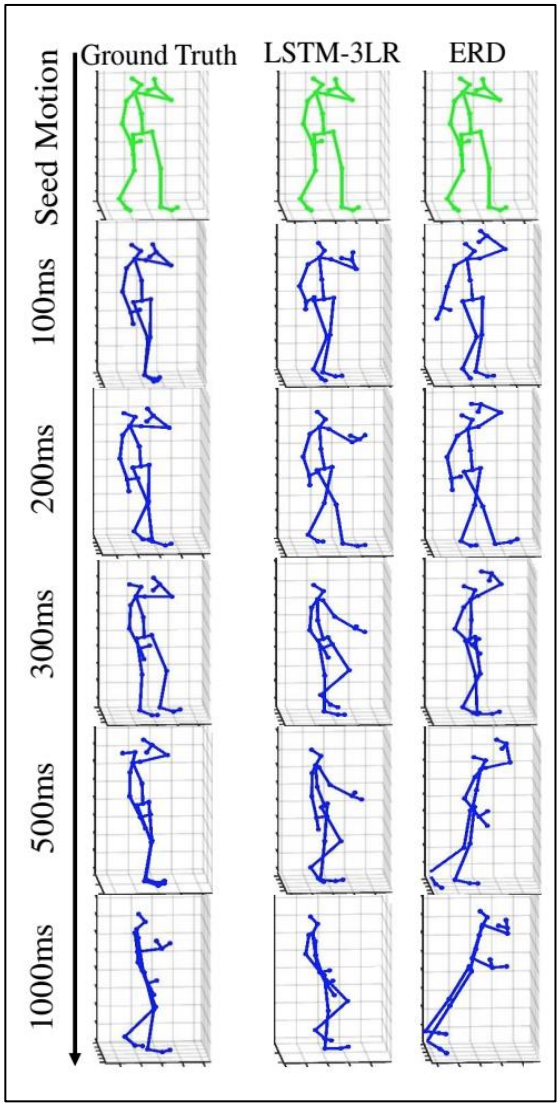


Figure **2-3**: Average frames at various time instances using LSTM-3LR and ERD

Figure 2-3, from A.Jain 2015, shows a comparison between LSTM-3LR and ERD frames. The LSTM-3LR networks eventually produce a viable mean pose while ERD drifts into unrealistic poses. Both architectures introduce gradient noise during training.

LSTM-3LR

In principle, a large enough RNN should be sufficient to generate sequences of arbitrary complexity [4]. In practice however, standard RNNs are unable to store information about past inputs for very long [16]. This reduces the ability of the model to generate stable long-term sequences. The problem (common to all conditional generative models) is that if the network's predictions are only based on the last few inputs, and these inputs were themselves predicted by the network, it has little opportunity to recover from past mistakes. [4] Since the variance of the error is continuously increasing, and each erroneous sample becomes the input in the next stage, the data quickly diverges and this is especially true in cases of real valued sequences.

A longer memory can increase stability. When the network cannot infer from the recent samples, it can use the samples that it received further in the past to create patterns in order to make predictions. Another remedy that has been proposed for conditional models is to inject noise into the predictions before feeding them back into the model, thereby increasing the model's robustness to surprising inputs. However, it is experimentally shown that a better memory is a more profound and effective solution. [4] LSTMs, using a multi-cell memory structure, have recently given state-of-the-art results in a variety of sequence processing tasks, including speech and handwriting recognition [15].

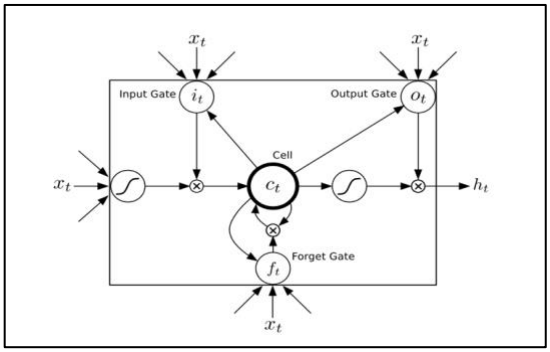


Figure **2-4**: LSTM cell illustrating various gates

Table 2: Error rates for LSTM-3LR - from A. Jain et al (2015)

LSTM-3LR	80ms	160ms	320ms	560ms	100ms
Walking	1.18	1.50	1.67	1.81	2.20
Eating	1.36	1.79	2.29	2.49	2.82
Smoking	2.05	2.34	3.10	3.24	3.42
Discussion	2.25	2.33	2.45	2.48	2.93

LSTM-3LR network, as shown by A. Jain et al [2] is prone to drifting errors which can be seen in the predicted frames, especially when acyclic actions are involved. This has been visualized in Figure 2-5 for the discussion activity at 560ms. The results shown in Table 2 are achieved by averaging error rates over 8 seed motion sequences for each activity on the test subject. The numbers indicate the 3D angle error between the forecasted frame and the ground truth This approach is similar to the one followed in the evaluation of ERDs. LSTMs have proven to perform better than ERD as they are not prone to drift errors and also show lower error rates for short term predictions.

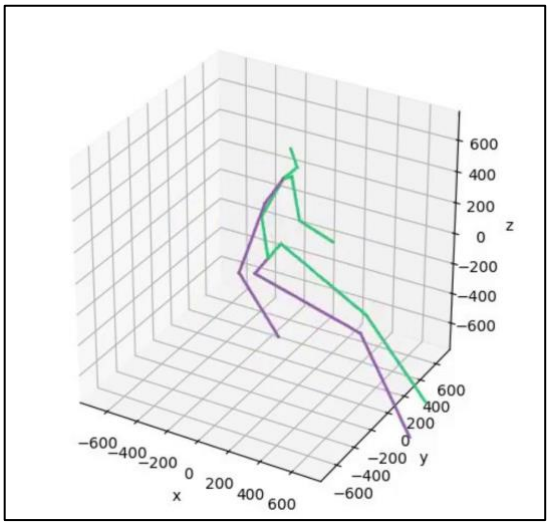


Figure **2-5**: Drifting error in LSTM-3LR for the discussion activity at 560ms

Structural- Recurrent Neural Networks

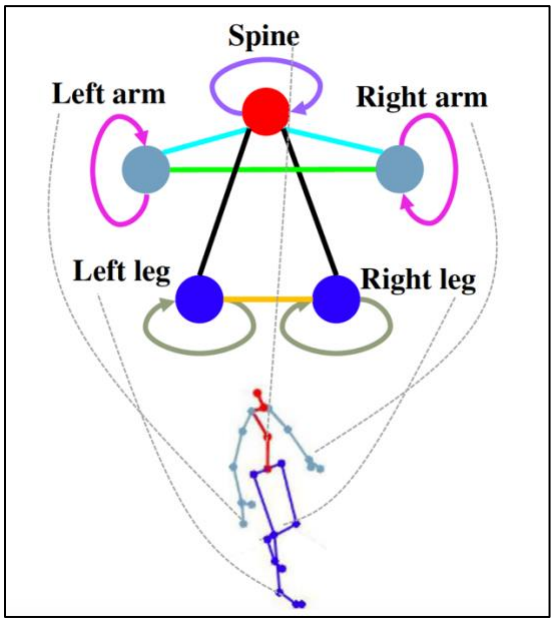


Figure **2-6**: Spatio-temporal graph of human body, by A.Jain (2015)

The human body has an innate hierarchical structure that arises from physical and biogenetic constraints. The spine/torso, defined in turn by the position of the head and pelvis, can be considered as a root unit. The graph shown above imposes a spatial constraint on the movement of joints. For example, the freedom of the elbow joint is constrained by the freedom of the shoulder and so forth. This information can then be fed into a graph-branch specific RNN and then combined with a common RNN to predict the overall motion.

Structural-RNN (S-RNN) architecture for human motion:

The S-RNN architecture follows the ST-graph shown in Figure 2-6. According to the ST-graph, the spine interacts with all the body parts, and the arms and legs interact with each other. The ST-graph is automatically transformed to a S-RNN. Similar to the previous algorithms,

noise is gradually added to the MoCap frames during training. This helps in keeping the forecasted motion close to the manifold of human motion.

Table 3: Comparison of error rates between SRNN, LSTM-3LR and ERD

Activity	Method	80ms	160ms	320ms	560ms	1000ms
Walking	ERD	1.3	1.56	1.84	2	2.38
Walking	LSTM-3LR	1.18	1.5	1.67	1.81	2.2
Walking	S-RNN	1.08	1.34	1.6	1.9	2.13
Eating	ERD	1.66	1.93	2.28	2.36	2.41
Eating	LSTM-3LR	1.36	1.79	2.29	2.49	2.82
Eating	S-RNN	1.35	1.71	2.12	2.28	2.58
Smoking	ERD	2.34	2.74	3.73	3.68	3.83
Smoking	LSTM-3LR	2.05	2.34	3.1	3.24	3.42
Smoking	S-RNN	1.90	2.3	2.9	3.21	3.23
Discussion	ERD	2.67	2.97	3.23	3.47	2.92
Discussion	LSTM-3LR	2.35	2.33	2.45	2.48	2.93
Discussion	S-RNN	1.67	2.03	2.2	2.39	2.43

From Table 3, It can be seen that the error rates in SRNN are much lower than that in ERD or LSTM-3LR. However, since there are several RNNs to be trained in SRNN, the computational requirements are very high. In this project, the goal is to choose the right algorithm to evaluate the frames generated by the GAN framework. This requires a comparison of the techniques based on multiple metrics.

Table 4: Ranking of techniques in the literature

Metric	ERD	LSTM-3LR	S-RNN
Overall Accuracy	2	3	1
Complexity	2	1	3
Long term prediction	3	2	1
Short term prediction	2	2	1
Input requirement	3	1	2

Table 4 ranks the algorithms with respect to several key factors involving motion prediction. In this work, the LSTM-3LR will be employed as the platform to evaluate the frames generated by the GAN network. This is due to the low complexity and already low requirement of input.

Chapter 3

Data- Human 3.6M

In the literature, all groups working on motion prediction with RNN [1,2,3,6] have trained and tested their models on the Human 3.6 M dataset by Ionescu et al.[5] This is currently the largest video pose dataset. The dataset consists of 15 activities, performed by seven professional actors and recorded from four calibrated positioned cameras. For each activity, subject, and camera viewpoint, there are two video sequences, each between 3000 and 5000 frames. Each activity features rich gestures, pose variations and several additions performed by the actors. For example, the walking activity includes holding hands, carrying a heavy load, putting hands in the pockets, looking around etc. The activities are recorded using a Vicon motion capture system that tracks markers on actors' body joints and provides high quality 3D body joint locations. 2D body joints locations are obtained by projecting the 3D positions onto the image plane using the known camera calibration and viewpoint. As in the literature, subject 5 will be used for testing and other subjects will be used for training. The raw data is comprised of 3D coordinates of 24 points of the human body and have about 4000 frames recording at about 50 frames per second.

The preprocessing of the data is done by Fragkiadaki et al. to convert the 3D coordinates into 3D joint body angles with the inclusion of degree of freedom information. This has been used by all groups since. The data is then standardized by subtracting the mean and dividing by standard deviation over all dimensions. This data has a sliding window of 5 frames which are fed into the LSTM network. This can be considered as times t0-t4 of the sequence. The prediction in the literature is over 10 frames. The output sequence in LSTM-3LR, SRNN and ERD are of 10 frames length and in this work, the frame rate will be kept the same to maintain consistency.

Chapter 4

Generative – Adversarial Networks

Mathematically, we think about a dataset of examples $x_1,\ldots,x_n$ as samples from a true data distribution p(x). In the example image, Figure 4-1, the blue region shows the part of the image space that, with a high probability (over some threshold) contains real images, and black dots indicate the data points (each is one image in a dataset). Now, the model also describes a distribution (green) that is defined implicitly by taking points from a unit Gaussian distribution (red) and mapping them through a (deterministic) neural network — the generative model (yellow). The network is a function with parameters θ, and tweaking these parameters will tweak the generated distribution of images. Our goal then is to find parameters θ that produce a distribution that closely matches the true data distribution (for example, by having a small KL divergence loss). Therefore, one can imagine the green distribution starting out random and then the training process iteratively changing the parameters θ to stretch and squeeze it to better match the real distribution.

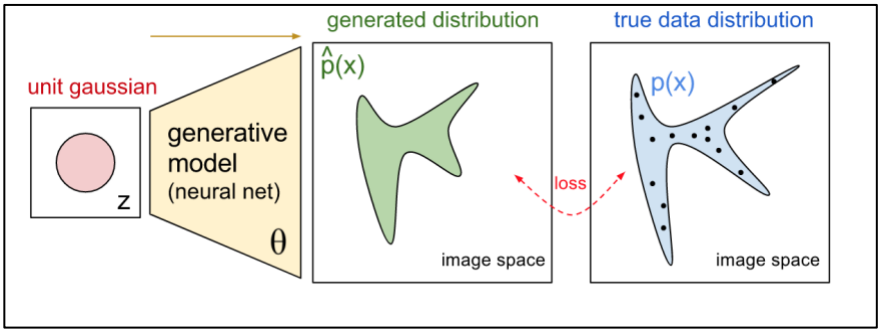


Figure **4-1**: Theoretical Representation of GANs

In the literature, GANs have been used for multiple purposes. The fundamental purpose of GAN is to fill in missing or unavailable data. This adversarial learning technique can be used to generate images from multiple image types [17], it can de-blur images, retrieve lost information in 3D point clouds, etc. In the original paper by Ian Goodfellow et al, they train the network to generate images of numbers which are discriminated as either synthetic or real. An alternative to MSE is the adversarial loss, as in the Generative Adversarial Network(GAN) (Goodfellow et al., 2014). This framework involves training a generator and discriminator in a minimax fashion. Successful extensions, including a conditional GAN (Gauthier, 2014; Mirza & Osindero, 2014) and a Laplacian pyramid of GANs (Denton et al., 2015), show its promise as a useful model for generating frames. The generator over several epochs learns to synthesize data which are extremely similar to the original training data. An example of GANs being applied to the MNIST dataset is shown in Figure 4-2.

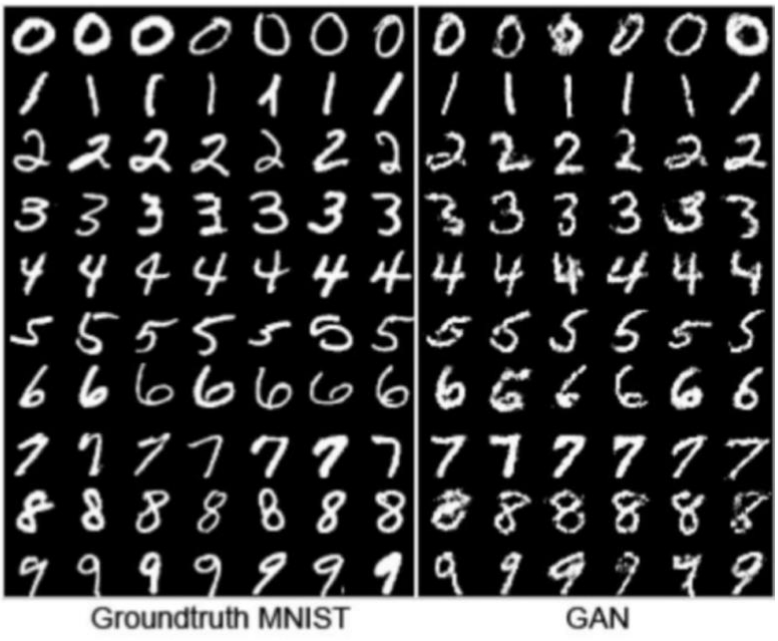


Figure **4-2**: Application of GAN on MNIST dataset [11]

Though GANs have shown immense success in the synthesis of various kinds of data, the training of the network itself is cumbersome, highly unstable and sensitive to changes in the hyper-parameters. Unlike traditional deep learning networks trained upon the scalar gradient descent landscape, the GAN network is expected to reach an equilibrium which cannot be theoretically predicted in advance. The function these networks try to optimize is an adversarial loss function that essentially has no closed form, unlike standard loss functions like log-loss or squared error. Thus, optimizing GAN loss function is very hard and requires a lot of trial-and-error regarding the network structure and training protocol.

Mode Collapse:

In GANs, the generator and discriminator are trained simultaneously, raising is possibility of mode collapse. When generator synthesizes a particular multimodal datum like an image or a motion frame, and the produced data is accepted (discriminator fails to classify it as fake), the generator might try to produce several more images or frames of the same type without actually learning the distribution of the mode. This can be fixed using a technique known as mini-batch training. Mini-batch gradient descent is a variation of the gradient descent algorithm that splits the training dataset into small batches that are used to calculate model error and update model coefficients. Implementations may choose to sum the gradient over the mini-batch or take the average of the gradient, which further reduces the variance of the gradient. Mini-batch gradient descent seeks a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. It is the most common implementation of gradient descent used in the field of deep learning.

Chapter 5

Predictive Generative Model

The aim of this project is to reduce the input requirements of current motion prediction techniques. In order to do this, the generation of synthetic frames is performed. The key idea is to use a GAN network to learn how to generate realistic synthetic frame for a given sequence. Evaluation of the proposed approach is done by calculating the mean error in the prediction of motion sequences for both synthetic and real frames. Then the total input requirement is calculated by estimating the total ratio of the real input frames to the number frames required to predict the motion for 160ms of prediction. Additionally, the generated synthetic frames are tested with the ERD algorithm to test the prediction error and compare against LSTM-3LR. Finally, the proposed approach is tested on a real-world video of pedestrian data and the prediction error is evaluated.

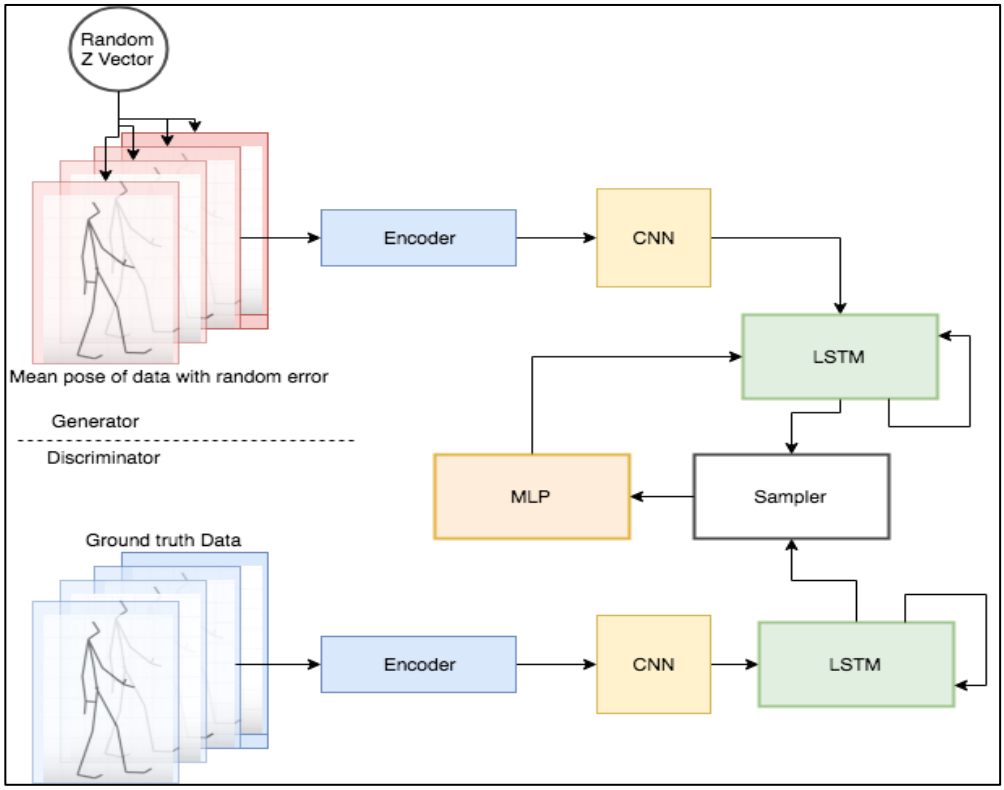


Figure **5-1**: Block Diagram of the proposed GAN Network

The process begins with the selection of a specific activity. If, for example, a walking activity is considered, the average mean frame of walking from 4 subjects is extracted and stored in memory. The input to the network is a sliding window of 8 frames at any given time. GANs are susceptible to a kind of failure known as mode collapse which occurs when either generator starts producing mean pose frames and does not learn or when the discriminator starts accepting frames with very low probability of being from the original data. To avoid this, it is common to perform class conditional batch processing in order to keep the variance in the input distribution high. The generator and discriminator networks are both trained simultaneously with the generator trying to synthesize motion frames and the discriminator MLP trying to reject these synthetic frames at each time step. Over time as the generator gets better at generating realistic frames, the discriminator also gets better at differentiating between real and fake frames.

Generator:

Given a particular activity sequence, the mean frame is taken from memory and added with a random vector *z* which is the same size as the input (24x3 points per frame). This is done for 8 frames and then the frames are fed to an encoder which converts the frames into a one-hot encoded sequence with length same as the input. The sequence is fed to a CNN to reduce the dimensionality in the feature space. The CNN consists of two layers of convolution, rectification, and max-pooling. The output of the CNN is fed to the LSTM network similar to work done by Hochreiter & Schmidhuber [16]. The LSTMs contain 3 layers with 1024 units in total. These units can be thought of as cells C_t which can be described mathematically as follows:

$$I_t = \sigma(W_{xi}X_t + W_{hi}H_{t-1} + b_i)$$
$$F_t = \sigma(W_{xf}X_t + W_{hf}H_{t-1} + b_f)$$
$$C_t = F_tC_{t-1} + I_t \tanh(W_{xc}X_t + W_{hc}H_{t-1} + b_c)$$
$$O_t = \sigma(W_{xo}X_t + W_{ho}H_{t-1} + b_o)$$
$$H_t = O_t \tanh(C_t)$$

Here, the input to the cell is X_t and the output is H_t. The input is accessed through input gate, I_t. F_t is the forget gate. The output depends on the current state of C_t and the past output of O_t. $W_{..}$ are the weight matrices, and σ is the elementwise logistic sigmoid function.

The output of the LSTM is the predicted sequence. The LSTM weight matrix requires a loss to train upon and the loss is obtained from the discriminator network. This loss is modeled on the adversarial loss function. The adversarial loss is given by the equation in Figure 4-2

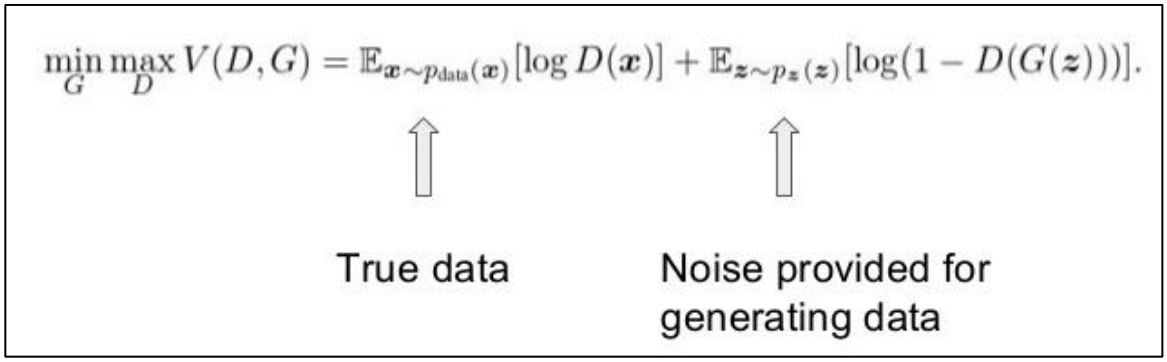


Figure **5-2**: Equation for Adversarial loss

Discriminator:

The discriminator network has purpose of differentiating between synthetic and real frames. In order to do this, an LSTM network similar to the generator is used. The input is however, the real ground truth frames coming from the H3.6M modified dataset. These frames are encoded and the dimensionality of their feature space is reduced with a CNN similar to the generator. However, the weights are different and the network is independent of the generator at this stage. The output of the CNN is fed to a LSTM network with 1024 units and randomized weights. The output of the LSTM is the true prediction sequence for that particular time window.

The sequences from Generator and the real data are randomly picked and fed to a Multi-Layer Perceptron (MLP) consisting of three FC layers with a sigmoidal read-out. The MLP acts as discriminator and differentiates between real and fake sequences. The discriminator outputs the probability of the proposed frame coming from the ground truth data. The discriminator is trained to maximize this probability when the frame is from the true distribution and minimize it when it is produced by the generator. The generator in return is trained to minimize this output probability by producing frame sequences closer to the ground truth data.

Assume the proposed frame synthesized by the generator is of the form G ($X^{i}_{1:t}$) and the discriminator function returns D (•, $X^{i}_{1:t}$) as the output. Given a mini-batch size of *n* sequences, the loss of the discriminator, $L_D^{(AL)}$ and the generator loss $L_G^{(AL)}$, have the form:

$$L_D^{(AL)} = -\frac{1}{2n}\sum_{i=1}^{n}[\log D(x_{t+1}^i, x_{1:t}^i) + \log(1 - D(G(x_{1:t}^i), x_{1:t}^i))]$$

$$L_G^{(AL)} = \frac{1}{n}\sum_{i=1}^{n}\log(1 - D(G(x_{1:t}^i), x_{1:t}^i))$$

At this point, there are two options for training the network. Either the generator can be trained to maximize log $(D(G(X^i_{1:t}),X^i_{1:t}))$ or the discriminator can minimize $\log(1-D(G(X^i_{1:t}),X^i_{1:t})$. Minimizing the discriminator log function tends to saturate the training early and accepts synthetic frames that are far from similar to the ground truth. For this reason, the first option is chosen. The loss obtained from the generator and discriminator is visualized in figure 5-3.

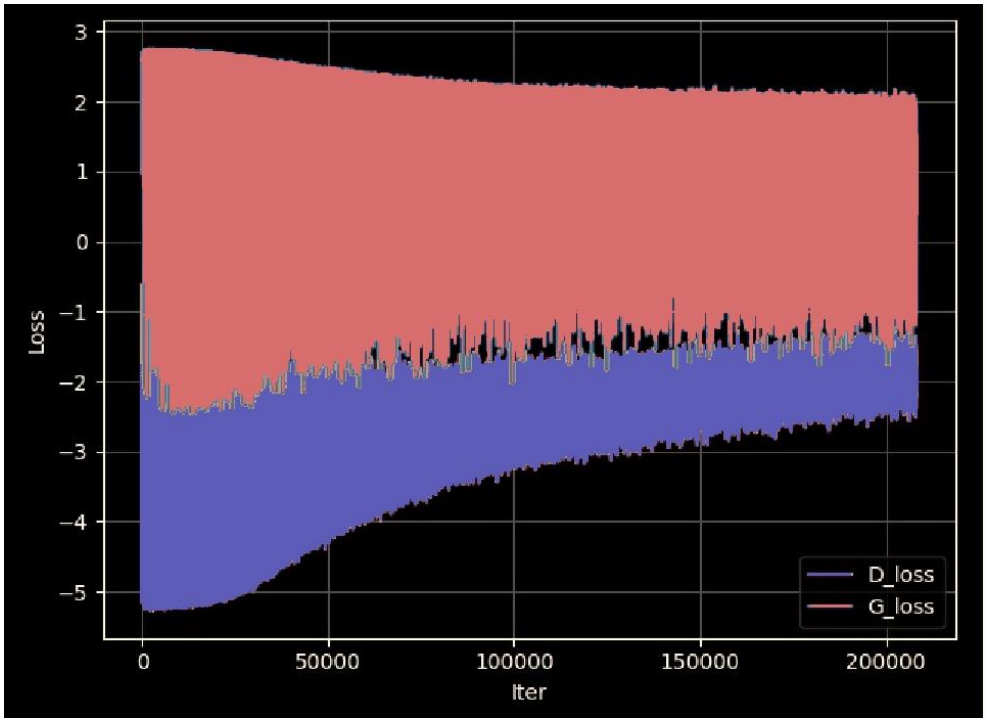


Figure **5-3**: Adversarial Loss visualization over 11 epochs for walking activity with subject 1

Upon successful training of the generator, the network simply samples from the noisy data (mean pose with random noise) and produces frames that have spatio-temporal probability distribution similar to that of the ground truth data. The generator being ready to produce

samples with a distribution similar to the data of the activity is exceptionally important as these frames are ready to be used for motion prediction completely on their own or as a mixture that is concatenated with the ground truth samples.

The function learnt by the generator is to transform the mean pose with random error to a frame as close as possible to the ground truth at that time instance. This is shown in Figure 5-4.

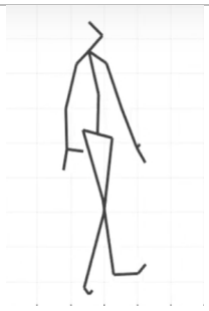
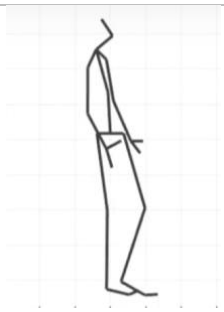
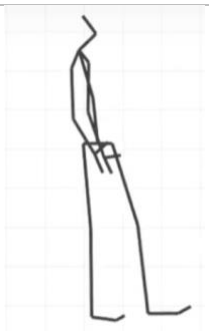
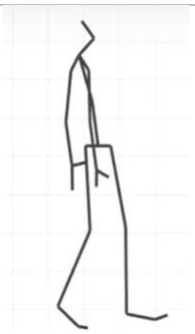
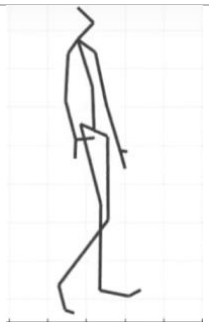
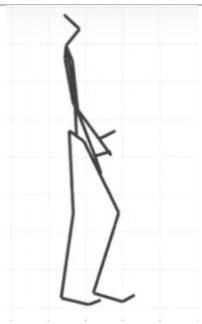
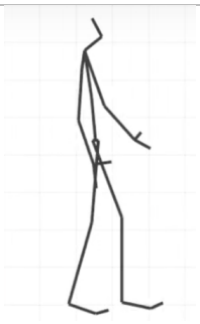
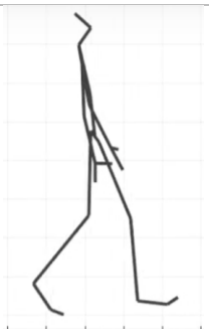
Time Instance	t0	t1	t2	t3
Ground Truth				
Mean Pose				
Synthetic				

Figure **5-4**: Sample frame comparison between real and synthetic data

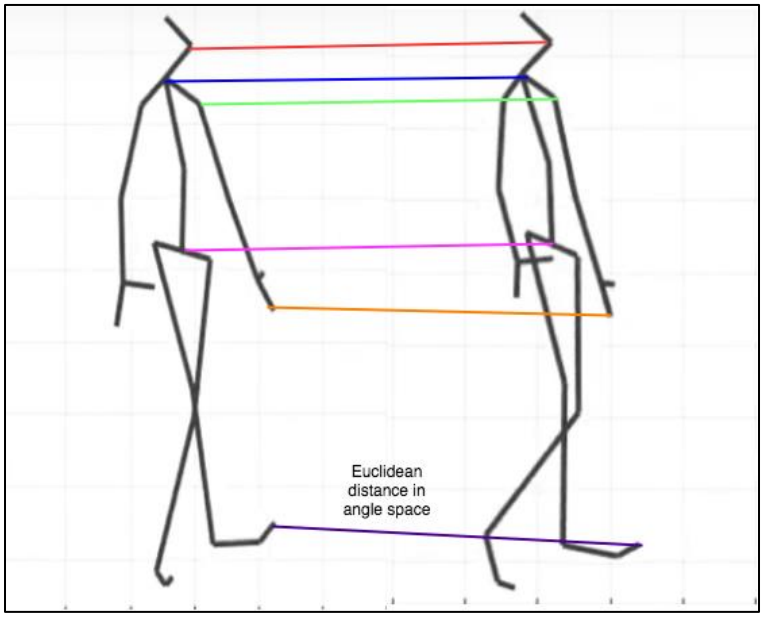


Figure **5-5**: Pairing joints to calculate error between two frames

In Figure 5-4, 4 random time instances t1-t4 are selected and the generated frames are compared. While the frames show visual similarity in terms of approximate locations of arms, legs and the overall direction of motion, it is essential to evaluate mathematically the error involved. The Synthetic frames shown are obtained after the completion of generator training. The errors shown in Table-5 are quantified estimates of how close synthetic frames can get to real data. This is done by pairing the corresponding joints between two frames in question and then calculating the Euclidian distances between the two 3x1 exponential map vectors. The average of these distances gives the frame error. 0 corresponds to complete match between frames and there is no upper limit to the error as distance can infinitely increase. However, practically this does not happen as the random vector z added to the mean vector also constricts the space the generator can sample from and thus restricts the overall error by a large margin. Times t0- t3 in the table are not related and not in order.

Table 5: Sample error calculation for 4 random time instances

Frame W.R.T Ground Truth	t0	t1	t2	t3
Ground Truth	0	0	0	0
Mean Pose	3.44	2.98	1.66	2.24
Synthetic Pose	1.31	1.07	1.52	2.35

Chapter 6

Experiments

Now that the frames are generated, they need to be evaluated using metrics used by groups in the literature so as to test the dependency of the techniques proposed by other groups on input length and type. There are 4 major experiments to be conducted in the procedure to evaluate the use of synthetic frames.

1. Train on Synthetic, Test on Real (TSTR) –ERD
2. Train on Synthetic, Test on Real (TSTR) – LSTM-3LR
3. Estimation of Input requirement
4. Application of Synthetic data to Real World tests
5. Use of multiple synthetic sequences for training

The GAN network generates a frame for each frame in the input real data. This means that at the end, when the generator is ready, there are equal number of real and synthetic frames. In the next step, where these frames are used for motion prediction, the synthetic frames themselves can act as ground truth. This however, will lead to a poor model, although with high accuracy. The correct technique would be to train the motion prediction algorithm on synthetic/ semi-synthetic data and then test on real data to evaluate. TSTR is also a method by which one can determine if a GAN network has simply memorized the training data. The TSTR technique is first tested on the ERD network which has the simplest architecture.

1. TSTR –ERD

The first experiment is to test the synthetic frames in an ERD setting which has, in the literature, shown minimal success for most activities. The ERD network was introduced by Frakiadaki et al [1]. The ERD network is among the weakest in predicting future frames especially when acyclic activities are involved. For this reason, ERD TSTR is performed only for the activity of walking.

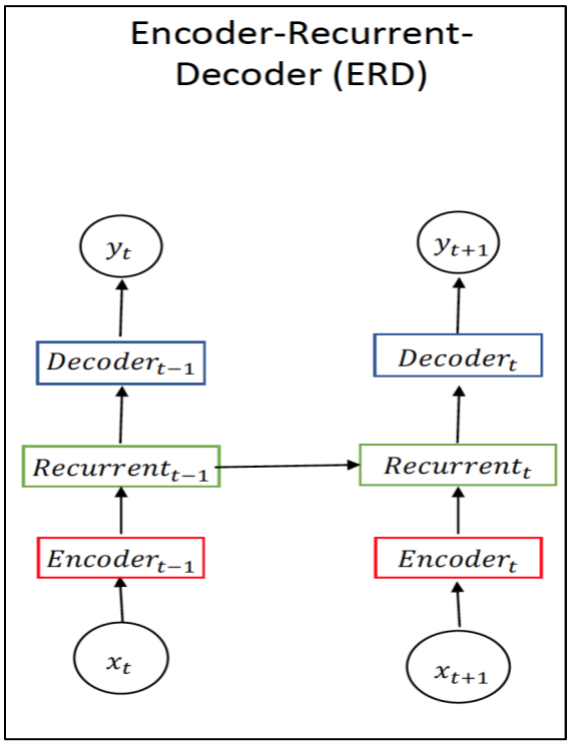


Figure **6-1**: ERD network as shown in Fragkiadaki et al (2015)

For evaluating the frames and the motion prediction technique on the whole, different quantities of synthetic data are considered for training. When no synthetic frames are considered, it is exactly like the original ERD work by Fragkidaki et al. When the network is trained entirely on synthetic data, the input ratio (number of synthetic frames in training/total number of frames in training) is 1. Lastly, when semi-synthetic data is considered, the input ratio can vary between 0

and 1. For this experiment, the input ratio is set to 0.45 by the method of trial and error. This translates to the fact that 45% of the training input is synthetic and 55% of it is the real data. The ratio is enforced by a sampler which at a given time instance chooses from either real or synthetic data sample space.

Table 6: Error rates for ERD with different input ratios

ms	Walking (R)	Walking (S)	Walking (M)
80	1.3	1.44	1.22
160	1.56	1.61	1.48
320	1.84	1.96	1.67
400	2	2.13	2.02

Table 6 shows the prediction error rates obtained at 80ms intervals. R is the column for real data, with 0 as the input ratio, S is the column with synthetic data with input ratio 1 and M is the mixture of synthetic and real data with input ratio 0.45.

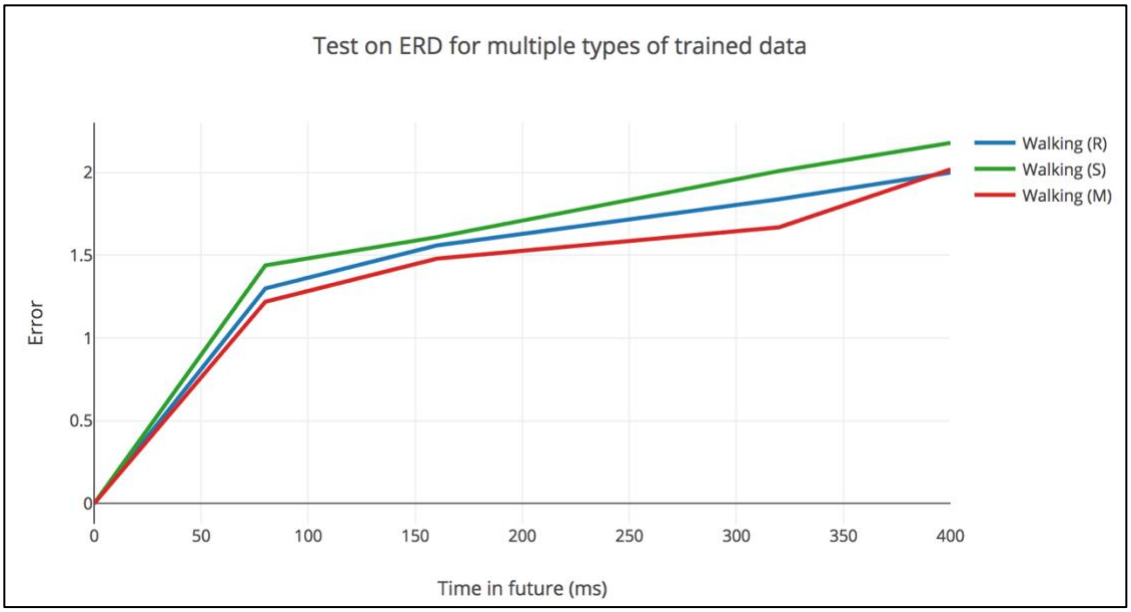


Figure **6-2**: Plot of error rates as a function of time

The error rates observed for ERD show that a mixture input of synthetic and real frames at a ratio of 0.45 has lower error rate for most of the near future predictions than that of the

original ERD working purely on real data. Although from Figure 6-2 there seems to be higher accuracy in using semi-synthetic data for motion prediction training, it has to be noted that the goal is only to get comparable accuracy while reducing the input requirement.

2. TSTR –LSTM 3LR

In this subsection, the LSTM 3LR network of Fragkiadaki et al [1] is fed entirely with synthetic frames and no real frames are used for training. The testing is done on the subject 5 of the H3.6M dataset and error is calculated as the mean frame-frame 3D angle distances. Since GAN performs poorly in generating multiple actions, the synthetic frames used and the frames at test time are all action specific.

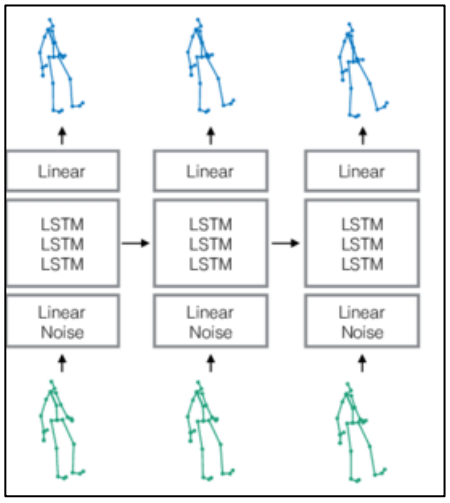


Figure **6-3**: LSTM 3LR architecture as illustrated in J. Martinez et al [3] (2017)

The green stick figures in Figure 6-3 represent the ground truth pose while the blue represent the predicted frame. This architecture was proposed by A. Jain et al [2] and consists of a 3 layered LSTM which performs sequence to sequence prediction. The input sequence of 24

points is fed to a linear encoder that converts the spatial information into binary information by the process of one-hot encoding and a noise vector which is scheduled beforehand is added to this. The noise vector is a gradient that is time dependent, i.e., At time t1, the noise added related to the noise added to the frame at time t2 by a linear function. The noise added is gradually increased over time to make sure the network experiences variation in the input. The sequence passes through 3 layers of LSTMs with about 1024 LSTM cells. The output of the third layer is a sequence which is one time step in the future. This can be decoded by passing it back to the encoder and the output is now 24 points of a frame. The predicted frame is fed to the input of the next time step to extract multiple frames and long temporal predictions. The results of this test are recorded below.

In this experiment, the ground truth input is replaced by real, synthetic or semi synthetic frames for each activity. The error is calculated and compared against the error obtained in the literature by using purely real sequences for training.

Table 7: Comparison of real and synthetic error rates with LSTM 3LR

ms	Walking (R)	Walking (S)	Eating (R)	Eating (S)	Smoking (R)	Smoking (S)	Discussion (R)	Discussion (S)
80	0.77	0.91	0.89	0.77	1.34	1.39	1.88	2.31
160	1	1.14	1.09	1.13	1.65	1.72	2.12	2.46
320	1.29	1.48	1.35	1.41	2.04	2.11	2.25	2.55
400	1.47	1.66	1.46	1.59	2.16	2.33	2.23	2.71

Table 7 shows how synthetic data performs against real data in predicting motion with an LSTM 3LR network.

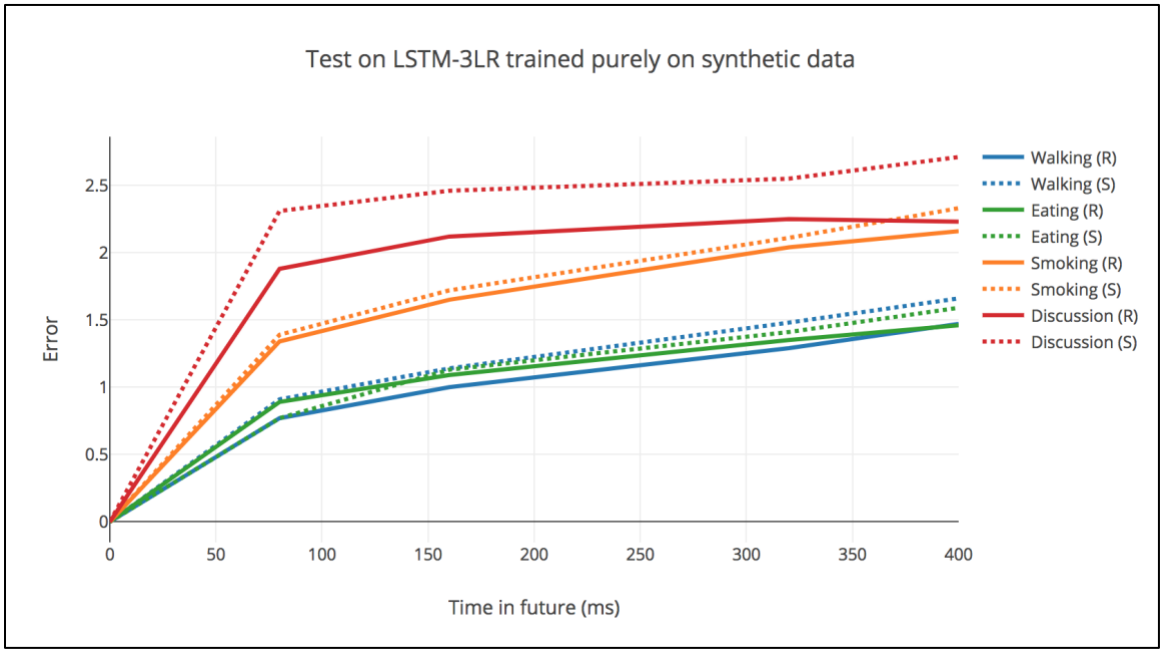


Figure **6-4**: Plot of error rates in LSTM 3LR, real vs synthetic

The error rates for frames predicted using real frames in training are the same as the error reported in literature by Martinez et al (2017) [3].

In the next test, the LSTM-3LR is fed the output of a sampler which samples from the synthetic and the real frames based on a predetermined ratio set by the user at training time. The input ratio is defined as the number of synthetic input frames to the total number of frames used for training. As an example, Figure 6-4 compares sequences with input ratio 0 (real) and 1 (synthetic)

Multiple ratios are experimented with and the test of input ratios is done in experiment 3. For this test, the ratio is 0.45 which translates to 45% of the training data being synthetic, which amounts to about 1890 to 2750 frames depending on the sequence length. This ratio has shown significantly lower error rate compared to other tested ratios.

Table 8: Error rates comparing real and mixture inputs with LSTM 3LR

ms	Walking (R)	Walking (M)	Eating (R)	Eating (M)	Smoking (R)	Smoking (M)	Discussion (R)	Discussion (M)
80	0.77	0.83	0.89	0.78	1.34	1.39	1.88	2.1
160	1	0.98	1.09	0.97	1.65	1.74	2.12	2.34
320	1.29	1.34	1.35	1.16	2.04	2.11	2.25	2.46
400	1.47	1.55	1.46	1.38	2.16	2.33	2.23	2.66

Figure **6-5**: Comparison of error rates for walking and eating activities

Figure **6-6** Comparison of error rates for smoking and discussion activities

The Mixture (M) of real and synthetic frames provided the best result for walking and eating at the ratio of 0.45. It can be seen that for eating, the error rate with the synthetic mixture is lower than the error rate for the system trained entirely on real frames. This is because the Synthetic frames produced by the GAN network easily mimic actions that are cyclic in nature. The GAN network has previously worked well in single point tracking and prediction, especially when the point follows a sine or cyclic motion.

3. Input ratio test

In this experiment, the ideal input ratio is calculated for different activities. This is performed by evaluating the error at 160ms for multiple ratios. This test is done for walking, eating, smoking and discussion activities using a LSTM-3LR platform parameter. Ir is the input ratio. Each column in the table below indicates the prediction error at 160ms for that activity at the given ratio.

Table 9: Comparison of error rates at different input ratios

Activity	Ir = 1	Ir = 0.8	Ir = 0.6	Ir = 0.5	Ir = 0.45	Ir = 0.25	Ir = 0
Walking	1.14	1.17	1.11	1.04	0.98	1.07	1
Eating	1.13	1.1	1.1	1.02	0.97	1	1.09
Smoking	1.74	1.81	1.75	1.74	1.72	1.66	1.65
Discussion	2.46	2.51	2.4	2.38	2.34	2.2	2.12

Table 9 shows the error rates at different input ratios. This is an important metric in deciding how much of the real input is required in making predictions.

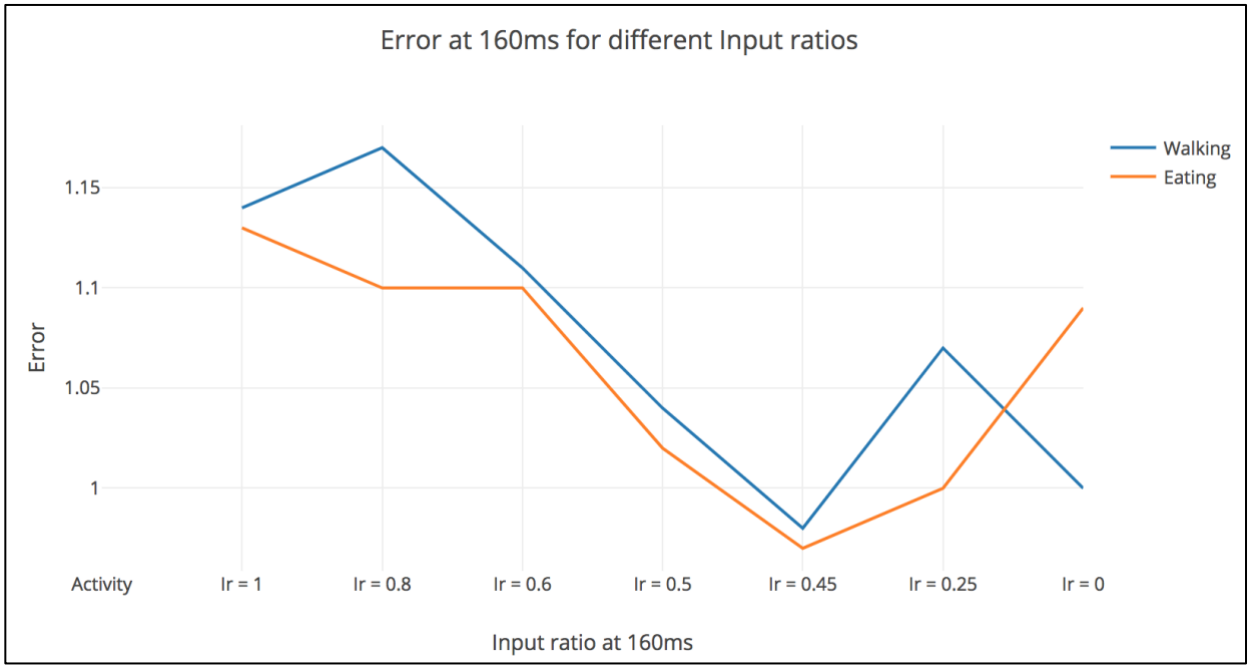


Figure **6-7**: Plot of error rates for cyclic activities at different input ratios

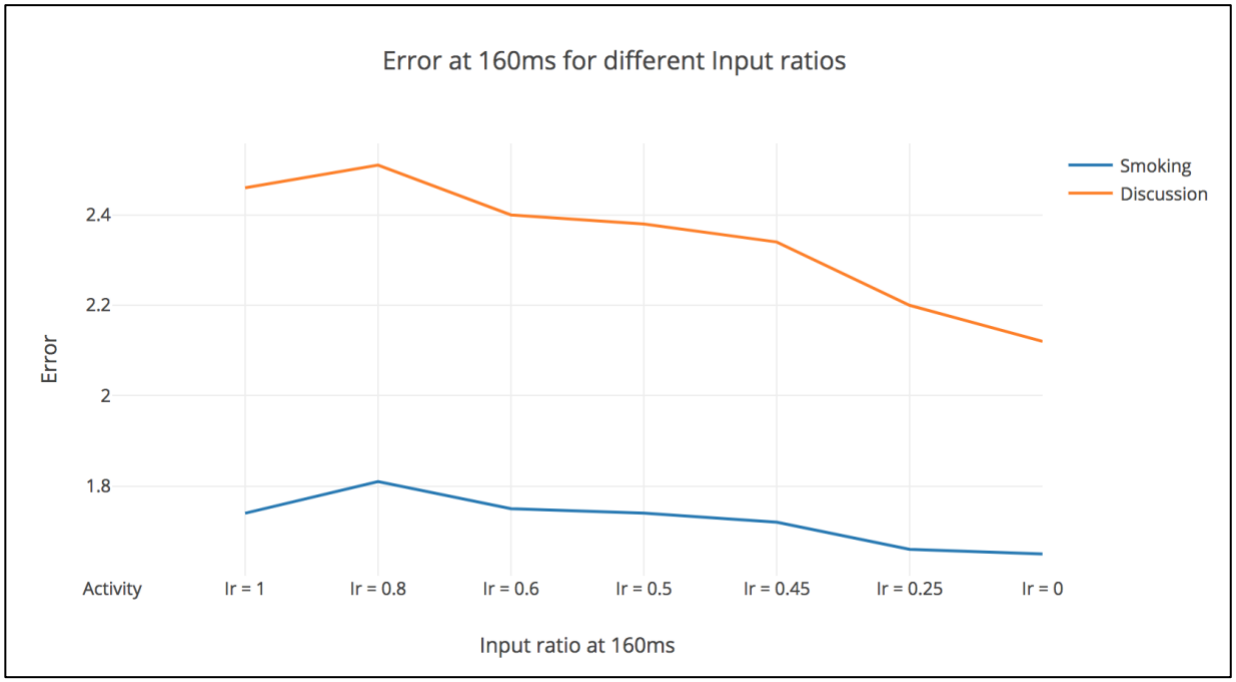


Figure **6-8**: Plot of error rates for acyclic activities at different input ratios

For cyclic activities, the ideal Input ratio is about 0.45 which indicates that about 45% of the input frames were synthetic and only 55% of the actual input was used to get the specific error rate. In cases of acyclic activities, the synthetic frames fail to accurately mimic the real data as learning their probability distribution is significantly difficult as many arm movements are random in these actions.

4. Using synthetic data in a real-world application

So far, all the tests performed were on the Human 3.6 M dataset which was obtained in a controlled setting with the use of VICON motion capture software and IR markers for identifying 3D joint locations. This, however, is far from the real-world situation where the location of Human body joints is hard to obtain due to noise and possibility of obstruction.

Table 10: Comparison of real-world data and H3.6M dataset

Metric	Real- world data	Human 3.6M
Visibility	2D	3D
Obstruction	Often	Rare, only few joints at a time
Speed	Video Captured at 30 fps	Data Captured at 50hz from multiple cameras and sensors ~ 1000fps
Noise	High	Almost 0
Length	Variable	Fixed at predefined length

For testing this application, the Human Activity Video (HAV) dataset is used. It has multiple subjects walking in a particular direction and the video is recorded using a monocular

camera similar to a CCTV or cameras found on autonomous cars. This dataset, however, does not have 3D joint locations or well-defined boundaries in order to directly process them in the LSTM 3LR network.

There are multiple approaches to this problem. One is to use a human pose estimation algorithm, which has been developed by multiple groups [13,14,15]. These algorithms usually find the location of a particular joint using a CNN network and then fit a 2D skeleton attaching it to the joint, after which other joint locations are optimized using a Euclidian error minimization technique. However, most algorithms only provide a 2D skeleton which cannot handle complex 3D motions. In the literature, groups have tried to extend 2D to 3D by adding a 3D viewpoint and applying epipolar geometry [12].

Mehta et al (2017) [14] have proposed a solution to this by using a model known as 2D Posenet. This is illustrated in Figure 6-9. The algorithms accept video frames and compares it similar actions performed in 3D by subjects in a setting similar to the H3.6M dataset.

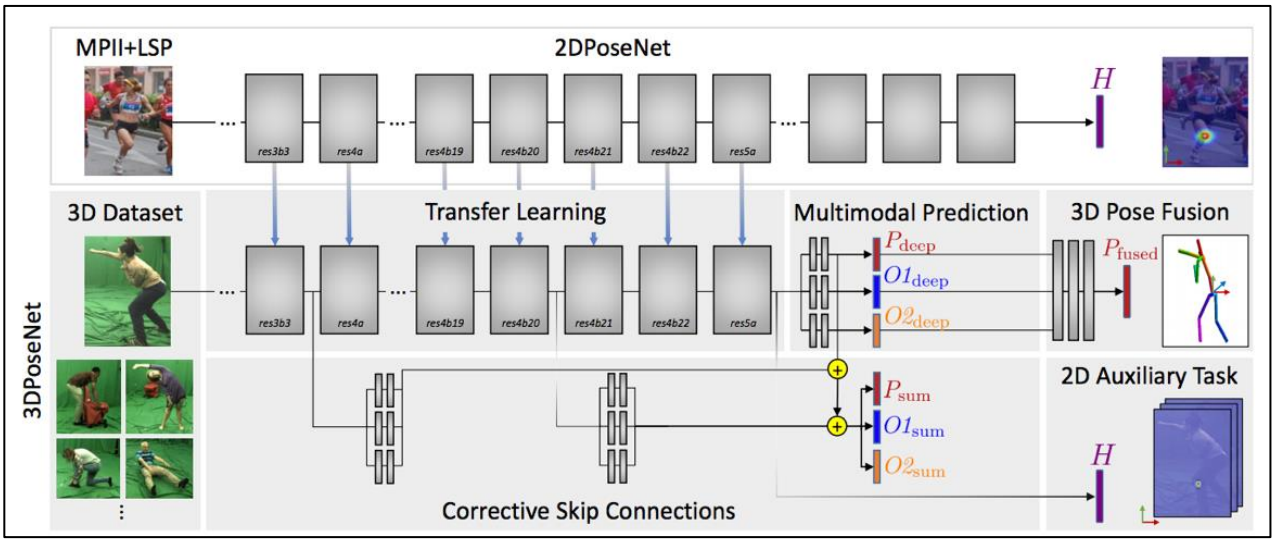


Figure **6-9**: 2D PoseNet as illustrated in Mehta et al (2017)

The 2D PoseNet first finds a heat-map of joints ‘H’ using the resnet architecture [14]. Simultaneously, the weights learnt for the heat map prediction of the 2D image are transferred to the second resnet structure which uses a 3D input and predicts the 3D location of the joints using the heat map and the previously learnt weights. The technique involves creating a 3D dataset for each of the 2D poses in the MPII Human Pose dataset and then performing transfer learning to match each 2D pose with a specific 3D pose followed by predicting the 3D locations. A sample visualization of a 3D skeleton superimposed on the 2D frame is shown in Figure 6-10.

Figure **6-10**: 3D skeleton superimposed on video frame

There are 10 subjects in the HAV-walking dataset and all perform the action of walking with different styles and speed. 4 subjects are used for testing. Videos of all 4 subjects are passed through the PoseNet framework and the output frames are then processed as regular test data on a LSTM-3LR platform and the error is calculated. It has to be noted that the sequences are much shorter than the sequences the network is trained upon. This is usually the case as the availability of long input real sequences is scarce.

The error in this data tends to be higher as the pose prediction network accumulates some degree of error as well. The testing is now performed with a network trained on real data, a

network trained on purely synthetic data and a network trained on their mixture. The errors are tabulated below.

Table 11: Error rates for real-world data

ms	Walking (R)	Walking (S)	Walking (M)
80	2.4	2.84	2.37
160	2.86	3.01	3.18
320	2.99	3.56	3.27
400	3.41	4.03	3.48

The test is performed by comparing the Euclidean distances between paired joint locations in the predicted frames and the ground truth (output of PoseNet). Figure 6-11 shows the plot of the error rates for walking with different input ratios. For the mixture training model, an input ratio of 0.45 was used.

Figure **6-11**: Error rates for real-world walking with different input ratios

Although the results indicate that synthetic data based system performs subpar compared to a system trained entirely on real data, it has to be noted that the error rates are comparable.

This indicates that the expensive real data can be replaced with synthetic data while incurring very small compromise in terms of accuracy.

5. Use of multiple synthetic sequences for training

The last experiment in this section is to test the variation of error due to different amounts of synthetic data used in training. In the earlier experiments, the total sequence length used at training was constant. However, in this experiment, the total training length is varied by introducing different amounts of synthetic data. The motivation behind the experiment is to train the network with as much data as possible while observing the errors at various time instances. The GAN network is made to generate multiple synthetic sequences and at the time of training the motion prediction network is given incrementally increasing amounts of synthetic data. The testing, like previous experiments, is performed on real data.

The total training data can be viewed as the sum of real data 'R' and synthetic data 'S'. In the first run, the network is given 'R+0S' which indicates that only real data was used for training. 'R+1S' indicates that the entire real data and one entire sequence of synthetic data was used for training. This process continues incrementally to R+4S. After each training, the motion prediction network is tested for prediction error with a real sequence of data. The error rates are as shown in Table 12.

Table 12: Error rates for varied training sequence length

ms	Walking (R)	Walking (M)	Walking (S)	Walking (R+1S)	Walking (R+2S)	Walking (R+3S)	Walking (R +4S)
80	0.77	0.83	0.91	0.71	0.69	0.82	0.98
160	1	0.98	1.14	0.92	0.95	0.99	1.06
320	1.29	1.34	1.48	1.19	1.26	1.31	1.38
400	1.47	1.55	1.66	1.36	1.41	1.38	1.44

The plot of the error rates for the activity of walking with different training sequences is shown in Figure 6-12.

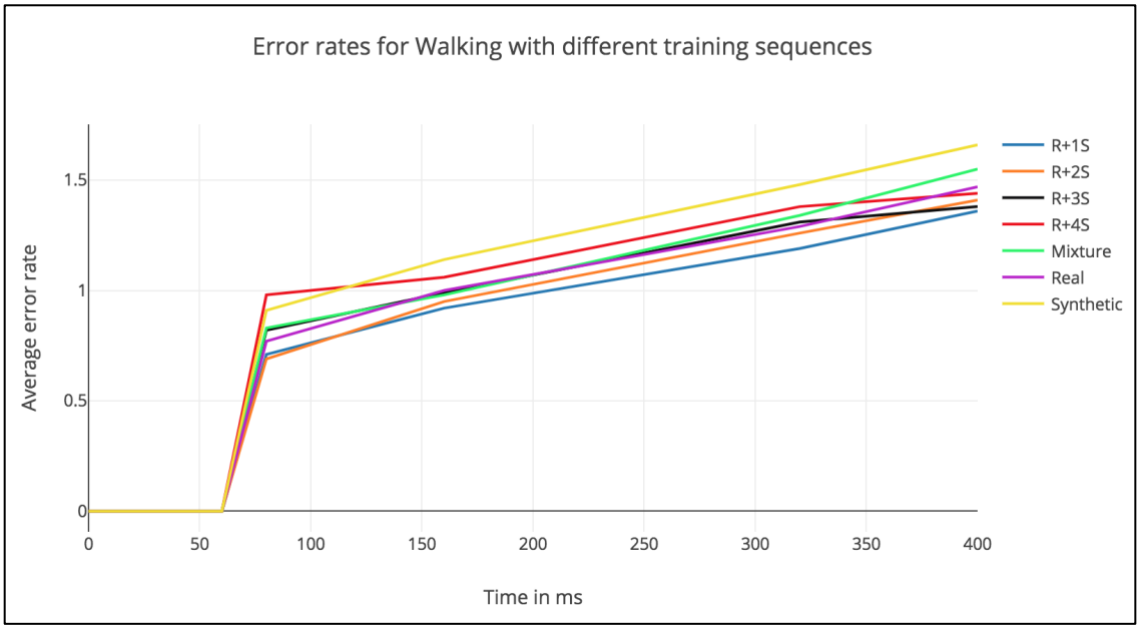


Figure **6-12**: Error rates for Walking activity with different training sequences

In the Figure 6-12, the 7 plots show the different error rates over time for different input lengths. If the available training sequence is of length ‘x’, then the plot ‘Real’ corresponds to the error rates of the network trained entirely on real data of sequence length ‘x’. ‘Synthetic’ corresponds to error rates of the network trained entirely on synthetic data of sequence length ‘x’ as well. ‘Mixture’ is the mixture of real and synthetic data at an input ratio of 0.45. The sequence length is maintained at ‘x’ for this case as well. In the next case, the network is trained on a sequence of length of ‘2x’ in which there is ‘x’ length of real data and ‘x’ length of synthetic data. From here on, the training input sequence length is increased by increasing the synthetic data used by ‘x’ length. In ‘R+2S’, the total training sequence length is ‘3x’ with ‘x’ length of real and ‘2x’ units of synthetic data. This is continued until ‘R+4S’.

The results show that initially, in the cases of ‘R+1S’ and ‘R+2S”, the network takes advantage of the surplus training data and performs better at the 80ms prediction. The error is lower than that obtained through the mixture model. However, as the synthetic frame content in training data increases, the prediction error increases. This can be seen in the cases of ‘R+3S’ and ‘R+4S’. In these cases, the dominant presence of synthetic data is simply distracting the network from the real pattern. It can be assumed that the overall variance in training increases with more synthetic data. This is due to the use of randomized vector ‘z’ used in creation of the synthetic frames. For the first two cases, the network was able to use the extra variance as tool to be prepared for sudden changes in the test data. However, with further increase in variance, the training data puts the motion prediction network past its bias-variance tradeoff point and thus increases the variance-based error due to over-generalization.

It also has to be noted that with longer sequences in training data, the motion prediction network performed better at long term predictions (400ms). All the test cases had lower error rates at 400ms when compared to networks trained on purely real, purely synthetic and mixture data.

Chapter 7

Summary and Discussion

The goal of this project was to design a system that would essentially reduce the dependency of motion prediction systems on long input sequences of real data. The idea originated from the fact that even the latest motion prediction techniques performed poorly when input lesser than 3500 frames was provided. In real world applications, most motion prediction systems do not have more than 2 to 3 seconds of input sequence to predict accurate short-term motion forecasts. This calls for a system that can handle low input data by filling in synthetic data to the required amount. The GAN based system does this by training once on the Human 3.6 M dataset after which it is potentially ready for motion prediction irrespective of input length. Although the initial goal was to train the network entirely on synthetic data and test on real world data, it was observed through the experiments that a mixture model with 45% synthetic and 55% real data had the lowest error rate and sometimes even lesser than a network trained entirely on real data sequence. The system showed the advantages of generating data rather than depending on real data sequences. The error rate achieved was lowest when large amounts of data was generated for training.

The cost of data is realized when motion prediction networks generate implausible frames with no clear pattern between joint locations. A machine learning system is only as good as its data. Low amounts of data usually would restrict the network learning and thus would perform poorly. GANs provide a beautiful solution to this constraint by sampling from noise and transforming the distribution of the samples to match the distribution of the original data. Thereby

generating unlimited amounts of data. Now systems can train on potentially large synthetic datasets and yet maintain accuracy.

Figure **7-1**: Sample predicted frame and real frame in future comparison

In Figure 7-1 it can be seen that the LSTM-3LR network predicts a very close match of the person walking and the prediction sample was obtained 160ms into the future. Predictions like these can make a large impact on how autonomous cars make decisions, on video based threat detection, and a lot of other applications where there is need for prediction of motion with low input.

Future Work:

The field of motion prediction and analysis is fairly new and has scope for major improvements. Applications using Predicted sequences usually fall into two categories:

1. Advanced decision making
2. Filling up data

In the first group, one can find applications like human robot interaction where the robot needs to predict human action to intuitively respond and make the physical/virtual interaction natural. Decision making is also crucial in autonomous vehicle systems that use prediction to understand pedestrian intentions. The reason sport videos are blurry at times is due to the fact that the motion of a player is hard to capture. In the future, deep learning based processing of videos can be done by predicting the action of players in advance and also performing video sequence prediction.

In the second group, the practical applications would be to fill up gaps on video sequences by simply generating synthetic frames wherever there is a loss of information. This is a powerful approach in the case of occlusion handling. In motion sequences, there is a lot of occlusion due to the restricted view of the camera. It can be assumed that the object occluded, will follow a similar pattern of movement as it did before or after occlusion. This assumption can help generate synthetic frames and fill up long gaps in the data.

The other major application of this system in the future is to transfer physical style of any activity of one person onto another person. GANs have already proven to perform exceptionally well in style transfer. They can learn from multiple datasets and morph the styles as per requirement. From a motion picture and graphics-animation perspective, the ability to imitate certain spatio-temporal features of certain objects or people can be very valuable.

Bibliography

[1] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, Jitendra Malik, "Recurrent Network Models for Human Dynamics", ICCV, arXiv:1511.05298, 2015.

[2] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. "Structural rnn: Deep learning on spatio-temporal graphs". CVPR, 2016.

[3] Julieta Martinez, Michael J. Black, Javier Romero "On human motion prediction using recurrent neural networks" CVPR, arXiv:1705.02445, 2017.

[4] A. Graves. "Generating sequences with recurrent neural networks". CoRR, abs/1308.0850, 2013.

[5] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. "Human3.6m: Large scale datasets and predictive methods for 3D human sensing in natural environments". TPAMI, 36(7), 2014.

[6] A. Jain, J. Tompson, Y. LeCun, and C. Bregler. "Modeep : A deep learning framework using motion features for human pose estimation". ACCV, 2014

[7] H. S. Koppula and A. Saxena. "Anticipating human activities for reactive robotic response". International Conference on Intelligent Robots and Systems, page 2071, 2013

[8] Wikipedia. Lagrangian and eulerian specification of the flow field

[9] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[10] G. W. Taylor, G. E. Hinton, and S. T. Roweis. "Modeling human motion using binary latent variables". NIPS, 2006.

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A.

Courville and Y. Bengio, “Generative Adversarial Networks”, NIPS, 2014.

[12] C. Esteban, S.L. Hyland, and G. Rätsch. “Real-valued (medical) time series generation with recurrent conditional GANs”. arXiv:1706.02633, 2017.

[13] Diogo C. Luvizon, David Picard, Hedi Tabia, “2D/3D Pose Estimation and Action Recognition using Multitask Deep Learning”, 2018

[14] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu,Dan Casas, Christian Theobalt, “VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera”, 2017

[15] Helge Rhodin, Jörg Spörri, Isinsu Katircioglu, Victor Constantin, Frédéric Meyer, Erich Müller, Mathieu Salzmann, Pascal Fua, “Learning Monocular 3D Human Pose Estimation from Multi-view Images” 2108

[16] Hochreiter, Sepp & Schmidhuber, Jürgen, “Long Short-term Memory”. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735. 1997

[17] Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros Berkeley AI Research Lab, “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”, 2017

[15] Bruno Stuner, Clément Chatelain, Thierry Paquet, “Handwriting recognition using Cohort of LSTM and lexicon verification with extremely large lexicon”, 2017

[16] Jonathan R. Stroud Matthias Katzfuss Christopher K. Wikle , “A Bayesian adaptive ensemble Kalman filter for sequential state and parameter estimation”, arXiv:1611.03835v1 , 2016

[17] Jack M. Wang, David J. Fleet, Senior Member, IEEE, and Aaron Hertzmann, “Gaussian Process Dynamical Models for Human Motion” Ieee Transactions on Pattern Analysis and Machine Intelligence, VOL. 30, NO. 2, 2008

[18] Siqi Nie, Ziheng Wang, Qiang Ji “A Generative Restricted Boltzmann Machine Based Method for High-Dimensional Motion Data Modeling” arXiv:1710.07831. 2017

[19] Elad Hazan, Karan Singh, “Cyril Zhang Learning Linear Dynamical Systems via Spectral Filtering”, arXiv:1711.00946. 2017.

[20] D. Lawrence, Neil. “Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data”. Advances in neural information processing systems. 2004.

[21] Graham W. Taylor, Geoffrey E. Hinton “Factored Conditional Restricted Boltzmann

Machines for Modeling Motion Style" International Conference on Machine Learning. 2009

[22] I. Akhter, T. Simon, S. Khan, I. Matthews, and Y. Sheikh. "Bilinear spatiotemporal basis models". ACM Transactions on Graphics, 31, 2012.

[23] BolaBola J.Z., Wang Y., Wu S., Qin H., Niu J. "Application of Hidden Markov Model in Human Motion Recognition by Using Motion Capture Data". In: Goonetilleke R., Karwowski W. (eds) Advances in Physical Ergonomics and Human Factors. Advances in Intelligent Systems and Computing, vol 489. Springer, Cham. 2016