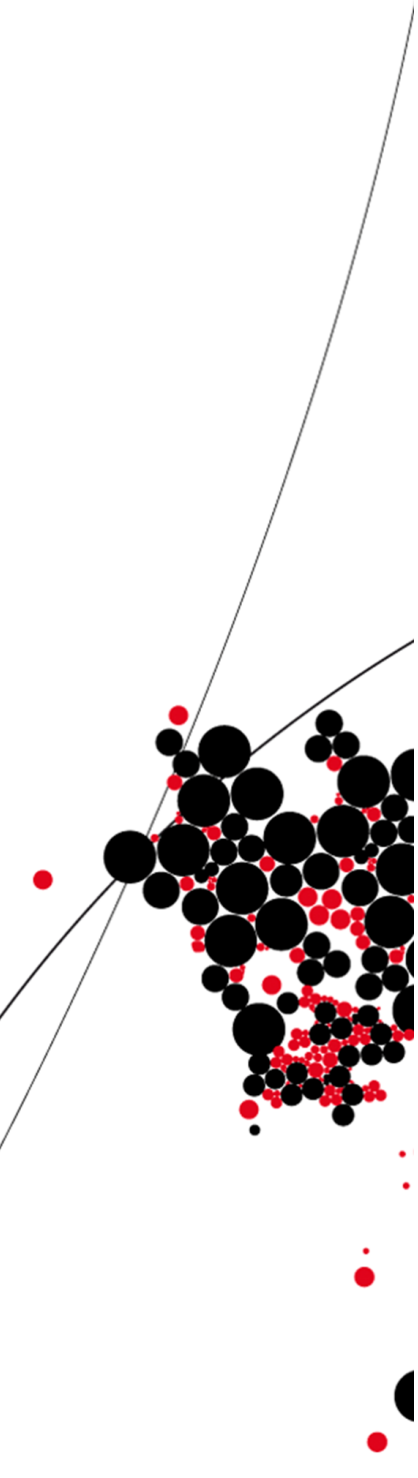
UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science

Generative Adversarial Networks of Missing Sensor Data Imputation for 3D Body Tracking

Xiaowen Song

Master Thesis
September 2020



Supervisors:
Dr. Mannes Poel
Dr. Ing. Aditya Tewari
Dr. Ing. Gwenn Englebienne

Data Management & Biometrics
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Acknowledgements

After six months of work, my final project is coming to completion. This period of time is precious and unforgettable to me. I would like to express my very great appreciation to those who were always there supporting and helping me.

Firstly, I want to thank Xsens Technologies B.V. and my colleagues at Xsens for giving me this opportunity and all the support and help along the way.

Secondly, I would like to thank my final project supervisors: Prof. Mannes Poel from University of Twente and Aditya Tewari from Xsens. Mannes, thank you for your patient guidance and constructive comments. The meetings with you allowed me to move forward firmly in the correct research direction of my thesis and maintain my enthusiasm for research. Aditya, thank you for your detailed guidance on my thesis and code. Without your help, my academic research and writing ability cannot be improved in a short time. The discussions with you brought me many new methods and solutions for my thesis. It was my pleasure to finish my thesis under your supervision.

Thirdly, I am particularly grateful for the support given by friends. In the two years since I came to the Netherlands, it was your support and company that made my life full of fun and motivation. At the same time, I would also like to thank my parents, thank them for their unconditional support and encouragement.

Last but not least, I would like to show my great gratitude to University of Twente and Northwestern Polytechnical University for the cooperation ”3+2” project, which provided me with the opportunity to complete my master’s degree in the Netherlands. This is an unforgettable and pleasant experience in my life.

Thank you!

Abstract

Human body motion tracking has important applications in many fields, not restricted to medical, biological science, virtual reality, sports and animation. While solving the problem of human motion tracking it is not always possible to obtain a large dataset without missing data or annotation. This creates challenges in developing algorithms that require such datasets. Moreover, reducing the number of sensors by generating data for these reduced sensors for motion capture can decrease the usage complexity. This thesis aims to design and evaluate efficient and precise machine learning models to impute the missing data for sensors used in body tracking solutions. Firstly, various traditional methods for data imputation and their shortcomings are introduced briefly. The characteristics of these methods that make them unsuitable for our tasks are then discussed. The human motion tracking datasets used in this thesis are obtained from sensors used in Xsens MVN Link inertial motion tracking system. Inspired by the traditional data imputation methods, we develop machine learning algorithms to deal with data imputation issues for human body motion tracking datasets. We first generate a model based on Hidden Markov Model (HMM) for data imputation in a time-series sensor signal. Further, an autoencoder based on convolutional and deconvolutional neural networks has been designed to impute the missing data in the motion tracking dataset. Finally, we investigate a Generative Adversarial Network (GAN) based method to solve the data imputation problem on the same dataset. The experiments are carried out with different lengths of missing data. The results of these three methods are evaluated and visualized. These algorithms are compared against two single data imputation methods: Mean Imputation and Zero Imputation. Dynamic Time Warping (DTW) and the Root Mean Square Error (RMSE) distance between the original dataset and the estimated imputed output are used for the evaluation of the three algorithms. The DTW measure shows that the proposed machine learning perform better than the two simpler single imputation methods. The DTW measure shows that proposed machine learning models produce better suited time series output as compared to Zero Imputation and Mean Imputation. HMM and autoencoder based models have better results on our datasets. Among the three algorithms, MisGAN based model achieves the best results. For the dataset with missing data of length 32 time frames, our MisGAN re-

duces the DTW value by 50.2% compared to Zero Imputation and reduces the DTW value by 50.4% compared to Mean Imputation. However, our models do not show obvious better performance than the two single imputation methods when evaluated using the RMSE measure. Through the analysis and visualization of these results, we consider that DTW is more suitable for analyzing the difference between time series data than RMSE. This research can be applied as solutions for data imputation for human motion tracking datasets, but further research needs to be conducted to make our models more suitable to human motion tracking datasets and to tune the parameters of models to improve the performance of them.

Contents

List of Figures

List of Acronyms

GAN	Generative Adversarial Network
GAMIN	Generative Adversarial Multiple Imputation Network
GAIN	Generative Adversarial Imputation Nets
WGAN	Wasserstein GAN
MCAR	Missing Completely at Random
MAR	Missing at Random
MNAR	Missing Not at Random
SVD	Singular Value Decomposition
DA	Domain Adaptation
VAE	Variational Autoencoder
DAE	Denoising Autoencoder
DTW	Dynamic Time Warping
RMSE	Root Mean Square Error
WGAN-GP	Wasserstein GAN with Gradient Penalty
EM	The Earth-Mover Distance
FID	Fréchet Inception Distance
ReLU	Rectified Linear Unit
HMM	Hidden Markov Model

Chapter 1

Introduction

In the motion tracking area, it is not always possible to obtain a large number of datasets that are without any missing data. Moreover, sometimes it is difficult to obtain a large number of fully labelled datasets. Failures happen when signals received from sensors are interrupted due to hardware or software malfunctions. Most algorithms for human body tracking use forward kinematics based on the human body skeletal model. The absence of sensors or data on body segments in the biomechanical chain makes the estimation using kinematics impossible. At the same time, algorithms that require motion tracking data usually rely on complete and labelled datasets, which emphasizes the integrity of labels and datasets.

From another perspective, in the human motion tracking area, it is considered desirable to reduce the number of sensors that collect motion data information [1] [2]. If the intention is to reduce the number of sensors for motion tracking, it can minimize the need for 'new and labelled data' while developing a sparse sensor solution for body motion tracking system. Additionally, a spare sensor based solution for body tracking reduces both the cost and complexity of use.

Many methods have been proposed to solve the missing data imputation problems. In general, these methods can be divided into two categories [3]:

- Using only the available partial data to estimate the parameters of the model
- Attempting to impute or predict the missing values with plausible values and then estimating the model's parameters

The disadvantage of the first category is that, with the remaining available data, the parameters of the model may not be estimated accurately. The second processing method is preferred, because it can use the imputed complete dataset for a more exhaustive and reasonable analysis as mentioned above.

The human body motion tracking datasets we processed in this thesis are collected by Xsens MVN Link systems. Xsens has two motion capturing systems: MVN

Link (wired) and MVN Awinda (wireless) 4.2. This thesis will focus on the MVN Link system. Xsens MVN consists of 17 (7 for lower body) inertial and magnetic motion trackers that capture full-body human motion in all environments [4]. This technology has found use in animation, sports, physical therapy, etc.

There are previous works to deal with data imputation. The most commonly used method is matrix completion [5] [6]. These methods require the matrix to meet the low-rank condition. The values of the entire matrix can be recovered from a limited number of entries. However, not all datasets satisfy the requirement of low-rank matrices. When the dataset is large, the complexity of the algorithm increases sharply. The classic imputation techniques, using the mean, median, or mode of observed data in the dataset to replace those missing data [3]. These methods are not precise in some situations, because the imputed dataset cannot reflect the real data distribution. Some other statistical based methods, like multiple interpolations and maximum likelihood estimation [7] are hardly scalable to large datasets. Compared to traditional statistical methods, machine learning techniques lead to statistically significant improvements in prediction and imputation accuracy [8]. In this thesis, we propose to apply more flexible and accurate methods to solve the data imputation problem on human body motion dynamic measurement during human movements.

As a method of exploring raw and unknown data, unsupervised learning is widely used as a method of machine learning algorithms. With the advancement of society and technology, the amount and complexity of data are rapidly increasing. Machines can find out unknown patterns in data without any form of training data or guidance with unsupervised learning [9]. These characteristics make unsupervised learning suitable for human motion dataset with a huge amount of data and sometimes suffering from missing data or labels. Meanwhile, artificial neural network algorithms are applied in various fields [10]. Neural networks can discover complex structures in high-dimensional data and extract different features [11]. The most important feature is their scalability, suitable for large datasets and flexible structural composition.

As one of the unsupervised learning and neural networks based methods, since the initiation of Generative Adversarial Networks (GANs), it has been extensively studied due to its huge application prospects in the image and visual computing, speech and language processing [12]. GANs have been proven to be a powerful machine learning tool in image data analysis and generation [13]. Many GAN based models are applied to data imputation for image processing. Compared to image data, time series data contains more information and data distribution is denser. GANs are rarely used to do data imputation for time series datasets. GAN based MisGAN is designed to impute image datasets [14]. In this thesis, we propose various supervised and unsupervised learning methods to deal with the issue of data imputation. To compare the MisGAN based method with other machine learning

based methods, we design experiments based on Hidden Markov Models (HMMs) and Autoencoders. After that, we conduct experiments to explore the feasibility and intuitive effect of applying MisGAN to impute human body motion time series datasets obtained by the Xsens MVN Link system. Finally, to study metrics to reflect the quality of correction for data imputation achieved using these aforementioned methods.

1.1 Research Goals

Based on the motivation, our goals are:

1. To study the previously used methods for data imputation and analyze their characteristics and limitations. Explain why these traditional data imputation methods are not suitable for human body motion data and the reason why choosing machine learning based methods.

2. To investigate machine learning algorithms that have been employed in the problems of data imputation. Discuss the possibility of applying these methods to the human body motion data imputation area.

3. To explore and develop algorithms selected in Research Goal 2, which help to compensate for the missing data in the human body motion datasets or to minimize the need for 'new and labelled data' for human body motion tracking datasets.

4. To develop metrics to report the quality of correction achieved using the aforementioned methods.

1.2 Outcomes

This thesis develops innovative machine learning based solutions for data imputation issues in the human body motion tracking area. It proves that: Traditional data imputation methods are not suitable for solving the problem of imputing human motion data. Our machine learning algorithms based data imputation methods can be applied to impute missing data or sensors in motion tracking datasets accurately. These imputed complete datasets can be further applied to related existing algorithms that deal with human motion tracking datasets.

1.3 Contributions

This section briefly presents the contributions of this thesis.

1. This thesis investigates previous mathematical methodologies that deal with data imputation and point out their limitations, which prompt us to figure out more efficient and suitable methods for our problem.

2. Machine learning based methods: HMM and autoencoder are developed to impute human body motion tracking data. These models are compared against MisGAN. MisGAN is applied to the imputation of human motion tracking data or sensors for the first time.

3. Dynamic Time Warping (DTW) and Root Mean Square Error (RMSE) are used for evaluation. DTW is more suitable for reflecting the difference between time series.

4. Comparisons are carried out among the aforementioned three methods. The experiments include the application of two single imputation methods in the meanwhile. It is demonstrated that those machine learning solutions perform better than two single imputation methods and MisGAN based framework has the best results. Under different lengths of missing data, MisGAN reduces the value of DTW to about half of the two single imputation methods.

1.4 Report Organization

In Chapter 2, previous works dealing with missing data and their limitations and features are discussed. Some machine learning algorithms applied to deal with data imputation issues in this thesis are presented. In Chapter 3, the missing data mechanisms used in the later chapters are introduced. Some basic concepts and potential training difficulties of GANs are discussed. At the same time, other machine learning based data imputation methods: autoencoders and HMMs are proposed. Chapter 3 then introduces a model called MisGAN, which is a GAN based framework that generates missing data distribution. It introduces a more reliable training method that uses the gradient penalty with Wasserstein GAN (WGAN). Finally, it discusses some metrics for the evaluation of the results of different methods. In Chapter 4, we first briefly introduce the human body tracking datasets used in the experiments. A HMM based model is generated to do data imputation with different missing rates. Then we develop a deconvolution and convolution neural network based autoencoder called Conv-AE to impute missing data. Finally, we present the architecture

of the neural networks applied in the MisGAN based framework. It then provides the methodologies of MisGAN based framework for our datasets. In Chapter 5, it first shows the results of applying HMM and Conv-AE on human body motion tracking datasets. Then we do some visualizations of data imputation. Secondly, we do experiments on MisGAN framework to explore the possibility of MisGAN method and visualize data imputation results. Moreover, these experiments are conducted under different model parameters (e.g. Different missing rate of datasets). Chapter 6 discusses the conclusions of the Research Goals and experiments. It formulates further research to be carried out in the future.

Chapter 2

Background

As discussed earlier, developing the capability to estimate the missing components of a dataset or data stream allows improvement in the performance of various data dependent tasks. The body tracking problem will also benefit from the introduction of data imputation methods. This chapter discusses some existing methods for solving the problem of data imputation and further explains the characteristics and shortcomings of these methods, which prompted us to find more accurate and reliable data imputation methods. Based on the previous findings, we develop some machine learning algorithm based methods to deal with missing data imputation issues.

Firstly, several traditional methods that solving data imputation are discussed in Section 2.1 and Section 2.2.1. The comparative advantages and shortcomings of these methods are introduced. Finally, we present some machine learning based methods that we applied in this thesis to deal with data imputation issues in Section 2.2.2 to 2.5.

2.1 Traditional Data Imputation Methods

In this section, we introduce various existing traditional methods used to deal with data imputation problems as well as their limitations and characteristics.

2.1.1 Matrix Completion

Matrix completion is a method of imputing in missing entries in a partially observed matrix. It aims to impute missing entries in an incomplete matrix with certain conditions. Low-rank matrices are the most commonly used assumption. In a low-rank matrix, each column of the matrix can be represented by a linear combination of a small number of basis vectors. The following part briefly introduces two specific implementation methods and their attributes.

Jian-Feng Cai et al. propose a novel method to complete a large matrix from a small subset of its entries. By applying various convex constraints, they recover the matrix with minimum nuclear norm [5]. The methods are used in recovering approximately low-rank matrix or unknown low-rank matrix from limited information. These methods can be applied in machine learning, control, or computer vision area. It can also be used to restore the missing data in a survey. In their experiments, they recovered several examples of 1000×1000 size matrices within 1 minute. Rahul Mazumder et al. develop their matrix imputation by replacing the missing elements in the incomplete matrix with those elements obtained from a soft-thresholded Singular Value Decomposition (SVD) [6]. Their methods fit a rank 95 approximation to the full Netflix training set in 3.3 hours in computing approximations of a $10^6 \times 10^6$ incomplete matrix with 10^7 observed entries [15]. They focus on matrix factorization and decomposition. However, matrix completion algorithms usually require matrices to satisfy the conditions of low-rank and are not suitable for every dataset. Moreover, these methods are computationally expensive and time consuming when applied to large datasets.

2.1.2 Single Imputation

Mean substitution, using the mean value of data. The advantage is that it does not change the overall sample mean. In addition, a single imputation can be implemented in a process of regression, and a regression model based on observable variables can be used to estimate missing data. But often overfitting occurs.

2.1.3 Multiple Imputation

For the Multiple Imputation, instead of replacing a single value for each missing element, the missing elements are substituted with a set of reasonable elements that contain the natural variability and uncertainty of the right values. Its purpose is not to recreate the missing data as close as possible to the real ones, but to deal with missing elements to achieve valid statistical significance. The advantage of Multiple Imputation is that it recovers the natural variability of missing data and contains the uncertainty caused by missing data so that effective complementary data can be obtained [7].

2.1.4 Maximum Likelihood

The first step in maximum likelihood is to construct the likelihood function. Getting the maximum likelihood is to find the parameter that makes the likelihood function

as large as possible. If there are missing values, then we can generate the joint probability that observation is just the probability of observing the remaining variables. The overall likelihood is the product of the likelihoods for all the observations. As mentioned above, the next step is to modify parameters in order to make the likelihood function as large as possible [16].

In the case of a large proportion of missing data, the maximum likelihood method may be difficult to converge, thus it is complicated. There are such datasets that the distribution and maximum likelihood of the observed dataset cannot be analyzed [7].

2.2 Machine Learning based Data Imputation Methods

The methods described above have many defects. For instance, they are not being suitable for large scale datasets, difficult to generate complex distributed data and insufficient accuracy of results, which urges us to seek more efficient ways to solve the problem. Machine learning algorithms can be used as the solution because it can assist missing data in uncertain scenarios by discovering the distribution in latent space. Below we briefly introduce some machine learning methods and later verify some of them.

2.2.1 K-Nearest Neighbours Imputation

K neighbors' estimation can be selected based on some distance metrics and their average or weight average values. The mean value of weight is related to the distance between the K neighbors and the missing data. The closer the distance, the greater the weight [17]. In other words, use the observed data in the missing data neighbors to impute those missing data [18]. This method suffers from the constraints that the complexity of the algorithm is high. It is easy to have large errors with the real value, and difficult to determine the value of K.

2.2.2 Hidden Markov Models

A Markov chain is a model that provides us information about the probabilities of sequences of random variables, states, etc. They are values taken from some set. With a Markov chain, there is an assumption that the state before the current has no impact on the states after the current state [19]. Based on the Markov chain, HMM is a method of assisting a sequence of observations with a series of hidden classes or hidden states that explain the observations. The key point of HMMs is that the

likelihood of the observations depends on the states of the system are hidden to the observer (e.g. part-of-speech tags in a text).

HMMs have a strong probabilistic framework for recognizing patterns in stochastic processes. HMMs are widely used in data analysis to predict and generate new data, such as speech analysis, image processing, etc. Moreover, HMMs are applied to stock sequence analysis recently and significant results have been obtained. The advantage of HMM can be summarized as follow [20]:

- HMMs have a strong statistical foundation
- HMMs are able to handle new data robustly
- Computationally efficient and easy to evaluate
- Predicting similar patterns efficiently

In this thesis, we refer to Nguyet Minh Nguyen's paper on applying HMM to stock price prediction [21]. We generate an HMM model similar to theirs and use it to the human body motion tracking dataset. Therefore, new data (missing data) can be predicted based on previous data. The detail of this process is discussed in Section 4.2.

2.2.3 Autoencoder

As an unsupervised machine learning method, autoencoders can be used to compress and extract data, remove noise from the data, etc. However, the models it generates are often vague and lack the authenticity and accuracy of the models generated by the GANs framework [22]. An autoencoder, shown in Figure 2.1 is a neural network with three parts: An input layer, an encoding block with hidden layers, and a decoding block with hidden layers. The purpose of this network is to reconstruct its inputs. Map input to code through the encoder and then map the code to the reconstruction of the original input. In effect, the encoder learns a good low-dimensional representation of the input data and the decoder component of the autoencoder learns to accurately recreate the data from the low dimensional representation. As a result, autoencoders are trained to minimize reconstruction errors (such as Root Mean Square Error).

Autoencoder can be applied for data imputation [23]. John T.McCoy et al. propose a recent deep learning technique, variational autoencoders (VAEs). It has been used for missing data imputation. Missing data in the original data can be recovered through the extraction and reconstruction of VAEs [24]. Haw-minn Lu et al. create a multiple imputation model using Denoising Autoencoders (DAE) to learn

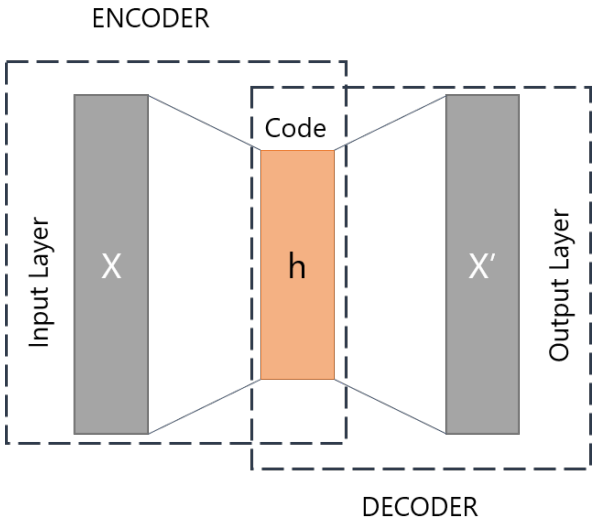


Figure 2.1: The structure of autoencoder

the representation of data, which is used to generated completed data for further processing [25].

2.3 Generative Adversarial Networks

With the development of machine learning technology, models based on deep learning provide us with novel ways to solve data imputation problems. These methods are usually easy to expand, which makes them suitable for datasets of different sizes and types, without requiring large training sets and test sets and they have features such as flexible structure. GANs can be used to generated distributions with a different dimension, which provides us with a solution to the problem of data imputation. The basic structure of a GAN is introduced below, and its application in data imputation will be discussed in Chapter 3.

GANs are unsupervised learning methods. Acquiring labeled data is a manual process that takes a lot of time. However, GANs do not require this labeling process. They can be trained using an unlabeled dataset as they can learn the internal representations of the dataset. GANs allow a deep learning model to capture the distribution of the input training dataset [26] and generate accurate results [27]. Ian J. Goodfellow et al. first proposed GANs. The specific workflow of GAN is discussed in Section 3.4.

2.4 Wasserstein GAN and Gradient Penalty

Martin Arjovsky et al. generate Wasserstein GAN to solve the delicate and unstable problems of GANs [28]. They direct their attention on the various way to measure how close the generated distribution and the real distribution are. They propose a new way to measure the distance between two distributions: The *Earth-Mover* (EM) distance, which is a measure of the distance between two probability distribution over a region [29]. To a certain extent, many issues such as generators' instability and gradient disappearance in GAN training are avoided by applying the new measurement of distance.

However, in the process of calculating the EM distance, unreasonable restrictions on the network may cause capacity underuse as well as exploding and vanishing gradient. Ishaan Gulrajani et al. propose a new method to clipping weight: penalize the norm of the gradient to the critical part about its input [30]. The algorithm is described in detail in Chapter 3.

2.5 MisGAN: a GAN for Missing Data

Various neural network based solutions [31] [32] [33] for sparse sensors tracking have been proposed. These solutions are based on supervised learning on real or synthetic data and do not exploit the available, albeit incomplete model information. Machine Learning based methods like Domain Adaptation (DA) [34], GANs can be employed to learn model information or to share information from one dataset to another. These methods can be used to extract useful knowledge from models or datasets and employ it to solve a task on another dataset with some mutually shared properties. GANs have been used for missing data imputation, which makes these methods suitable for sparse sensor based solutions.

Jinsung Yoon et al. propose a novel method for data imputation based on GANs called GAIN [35]. The GAIN imputes the unobserved part based on the available data, which indicates that the GAN-based structure can be used to impute data on incomplete datasets. Based on the previous work [35] [28], Li S et al. generate a GAN based framework MisGAN to learn the complex and high dimensional distribution of incomplete datasets [14]. The training process follows the WGAN-GP method mentioned in Section 2.4.

From the results of their paper [14], compared with some previous methods (e.g. GAIN), MisGAN has obtained significantly better results when imputing missing data in image processing. Inspired by it, we want to explore how to deal with data imputation problem using MisGAN framework in human body motion tracking datasets. It is further discussed in Chapter 3.

Chapter 3

Related Work and Definitions

Earlier we discussed some classical methods for solving data imputation problems. In this chapter, we begin with a discussion of three kinds of missing data mechanisms' characteristics in Section 3.1. We then focus on HMM, a supervised learning based generative model, which can be applied to data prediction in Section 3.2. In Section 3.3, we talk about a convolutional and deconvolutional neural network based autoencoder Conv-AE. We then present MisGAN, a neural network-based data imputation method. Section 3.4 presents GAN and Wasserstein GAN with Gradient Penalty (WGAN-GP). After that, Section 3.5 introduces the novel approach MisGAN taken by Steven Cheng-Xian Li and Bo Jiang and Benjamin M. Marlin [14], which uses WGAN-GP based training strategy to generate distributions and imputation of missing data. Afterwards, MisGAN will be further improved to be applied to deal with data imputation of the body tracking problem mentioned in Chapter 1. Finally, the metrics applied to evaluate the performance of various methods are presented in Section 3.6.

3.1 Missing Data Mechanisms

In order to deal with missing data, we are concern with the missing data mechanisms. Especially whether the value of missing data is related to the underlying value of the variables in the dataset. The nature of the dependencies in these mechanisms is crucial for choosing missing data methods. Some methods of data imputation require special conditions for missing data mechanisms, which will be discussed in detail when introducing these methods in this Section. Literature about missing data theory describes three main mechanisms [8]. Among the three missing mechanisms, we mainly focus on the first two forms.

Here we start to give some definition of missing data mechanisms. Let $Y = (y_{ij})$ denote the dataset without missing data. y_{ij} is the value of the variable Y_j for subject

i. Let $M = (m_{ij})$ denote the missing data indicator matrix, such that $m_{ij} = 1$ if y_{ij} is missing and $m_{ij} = 0$ if y_{ij} is not missing. Thus, the matrix M defines the pattern of missing data.

Missing Completely at Random

As mentioned above, we define the complete data $Y = (y_{ij})$ and the missing data indicator $M = (m_{ij})$. The mechanism of missing data can be formally defined by a conditional distribution $f(M|Y, \phi)$, where ϕ denotes unknown parameters. If the missing situation does not correlate with the values in dataset Y, then:

$$f(M|Y, \phi) = f(M|\phi) \ for\, all\, Y, \phi. \tag{3.1}$$

A missing data matrix is said to follow the Missing Completely at Random (MCAR) mechanism. Note that under this condition, it does not mean that the pattern of missing data is random, but the missing data does not depend on the dataset Y.

Missing at Random

Let denote $Y = [Y_{obs}, Y_{mis}]$, where Y_{obs} denotes the observed data in dataset Y and Y_{mis} denotes the missing data according to the missing data indicator $M = (m_{ij})$. The second missing data mechanism has fewer restrictions than the first mechanism. The missing data in the dataset Y is only related to the observable data Y_{obs} and does not depend on the missing data Y_{miss}. The Missing at Random (MAR) can be formally defined as follow:

$$f(M|Y, \phi) = f(M|Y_{obs}, \phi) \ for\, all\, Y_{mis}, \phi. \tag{3.2}$$

Missing Not at Random

If the distribution of M is related to Y_{mis}, then this mechanism is called Missing Not at Random (MNAR).

3.2 Hidden Markov Models for Data Prediction

Nguyet Nguyen proposes a HMM for stock price prediction [21]. As mentioned in Section 2.2.2, HMM is a generative probabilistic model. The system is considered to be transitioning in a certain finite number of states. The state transition can be defined by a matrix of state transition probabilities.

Consider A_t is the value of one element in a certain state and S_t to be the state on time frame t, which can be one of the assumed states. Then define some terminologies that are used to generate HMMs [21]:

- Number of observations: T
- Observation Sequence: $O = o_1 o_2 \dots o_T$, a sequence of T observations
- Number of states: N
- States: $Q = q_1, q_2, ...q_N$, , a sequence of N states
- State transition matrix: $A = [a_{i,j}]$, which reflects the probability of transition from s_i to s_j. (s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$)
- A sequence of observation likelihoods: $B = b_i(o_t)$ ($\sum_t b_i(o_t) = 1$). Each expressing the probability of an observation o_t being generated from a state i
- An initial probability distribution over states π_i, indicates that the probability that the Markov chain will start in state i: $\pi = \pi_1, \pi_2, \dots, \pi_N$ ($\sum_{i=1}^{n} \pi_i = 1$)

Hence the HMM can be represented as:

$$\lambda = (A, B, \pi).$$

Moreover, a hidden Markov Model has two simplifying assumptions. Firstly, the probability of a certain state only depends on the previous state of it.

$$P(q_i \mid q_1 \dots q_{i-1}) = P(q_i \mid q_{i-1}) \tag{3.3}$$

Secondly, "the probability of an output observation o_i depends only on the state that produced the observation q_i and not on any other states or any other observations" [19]. Thus we have:

$$P(o_i \mid q_1 \dots q_i, \dots, q_T, o_1, \dots, o_i, \dots, o_T) = P(o_i \mid q_i) \tag{3.4}$$

With these definitions and assumptions, we can build up a specific HMM for data prediction of human motion tracking data.

3.3 Working Principle of Autoencoder

In this section, we explain the possibility to apply autoencoders to recover the datasets with missing data. Autoencoder can be applied for data imputation [23]. An autoencoder is an unsupervised learning model.

Autoencoder is usually divided into two parts, namely encoder, and decoder. First, the encoder is used to encode the input data, and then the decoder is used to decode the encoded inputs. The purpose is to reduce the reconstruction error between the generated data and the original data as well as to find a low-dimensional representation of the input data. The basic structure of the autoencoder includes two parts: the encoder and the decoder. They are written as,

$$\begin{aligned} \phi &: \mathcal{X} \to \mathcal{F}, \\ \psi &: \mathcal{F} \to \mathcal{X}. \end{aligned} \tag{3.5}$$

Generally, the autoencoder is a neural network with more than one layer, but the basic working principle is the same as that of a single hidden layer autoencoder. Suppose in the simplest case there is only one hidden layer, we have:

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{3.6}$$

where $\mathbf{x} \in R^d = \mathcal{X}$ and $\mathbf{h} \in R^p = \mathcal{F}$ from Equation 3.3. σ is an activation function in neural networks. $\mathbf{W}$ is a weight matrix and $\mathbf{b}$ is a bias vector. The decoder maps $\mathbf{h}$ to the reconstruction of $\mathbf{x}'$, which has the same shape as $\mathbf{x}$:

$$\mathbf{x}' = \sigma'\left(\mathbf{W}'\mathbf{h} + \mathbf{b}'\right). \tag{3.7}$$

Autoencoders are trained to minimise reconstruction errors (such as squared errors). In order to minimize the difference between the data reconstructed by the autoencoder and the original data:

$$\mathcal{L}\left(\mathbf{x}, \mathbf{x}'\right) = \|\mathbf{x} - \mathbf{x}'\|^2 = \|\mathbf{x} - \sigma'\left(\mathbf{W}'(\sigma(\mathbf{W}\mathbf{x} + \mathbf{b})) + \mathbf{b}'\right)\|^2. \tag{3.8}$$

The autoencoder obtained after training on the complete dataset has the ability to restore the original data. At this time, the dataset with missing data is used as the input of the trained autoencoder. Then the trained autoencoder can output the imputed dataset.

3.4 Data Imputation of Minimax Optimization with GAN

As mentioned before, GANs can be used to generated distributions with a complex dimension and provides us with a solution to the data imputation issue. The basic structure of a GAN is introduced below and its application in data imputation will be discussed in this section.

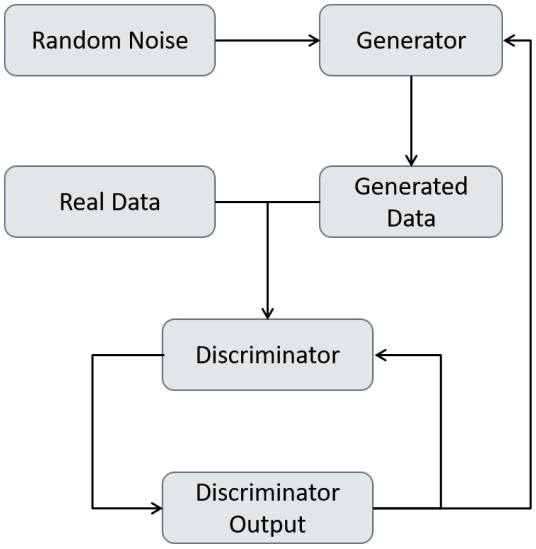


Figure 3.1: The structure of GANs

This workflow of this framework is shown in Figure 3.1. The arrows in this workflow represent the outputs or inputs of different modules. There are two parts in this framework: a generator G and a discriminator D. First use random noise (e.g. coming from Gaussian distribution) as the input of G. G is used to capture the distribution of real data by generating generated data. Generated data and real data are used as input for D. D is used to judge whether the input data comes from the real distribution or G. It distinguishes between real data and generated data as much as possible. The generator G minimizes the gap between the generated data and the real samples as much as possible. The most ideal state is that the discriminator cannot discriminate between the generated data and the samples from the real distribution. Fix G while training D, and vice versa. Take turns to train G and D until the desired results are obtained.

The GANs training strategy can be defined as:

$$\min_G \max_D \underset{\boldsymbol{x}\sim P_r}{E}[\log(D(\boldsymbol{x}))] + \underset{G(z)\sim P_g}{E}[\log(1 - D(G(\boldsymbol{z})))], \tag{3.9}$$

where P_r is the real data distribution and P_g is the distribution from G. x is real data and $G(\boldsymbol{z})$ is data generated by G, where $\boldsymbol{z}$ is random noise. We train D to maximize the possibility of correct classification of training examples and G generated samples. The strategy is to let G and D play a two-player minimax game with Equation 3.9.

3.4.1 Wasserstein GAN and Gradient Penalty

The purpose of training the GAN is to make the distribution P_g generated by the generator closer to the real data distribution P_r. In actual operation, the distribution generated by the generator is close to the distribution of real data, that is, the process of maximizing the value of Equation:

$$\underset{\boldsymbol{x}\sim P_r}{E}[\log(D(\boldsymbol{x}))] + \underset{G(z)\sim P_g}{E}[\log(1 - D(G(\boldsymbol{z})))] \tag{3.10}$$

is equivalent to maximizing the following equation based on Jensen-Shannon Divergence:

$$-2log2 + 2JS(P_r||P_g). \tag{3.11}$$

Jensen-Shannon Divergence is defined as follow:

$$JS\left(P_r\|P_g\right) = \frac{1}{2}KL\left(P_r\|\frac{P_r + P_g}{2}\right) + \frac{1}{2}KL\left(P_g\|\frac{P_r + P_g}{2}\right). \tag{3.12}$$

KL in the equation refers to the *Kullback-Leibler* (KL) divergence, defined as:

$$KL\left(P_r\|P_g\right) = \int \log\left(\frac{P_r(x)}{P_g(x)}\right) P_r(x)dx \tag{3.13}$$

Since most of these distributions that need to be generated by the generator are low-dimensional manifold distributions in high dimensions, the generated model and the true distribution's support do not have a non-negligible intersection [28]. This fact results in Equation 3.11 equal to a constant: $log2$, which means that the KL divergence is not defined, which causes the vanishing gradient of generator G [36].

To be able to measure the distance between two distributions that do not overlap. Martin Arjovsky et al. generate Wasserstein GAN to solve the delicate and unstable problems of GANs [28]. They direct their attention on the various way to measure how close the generated distribution P_g and the real distribution P_r are. In other words, to define better measures for distance or divergence $\rho\left(P_g, P_r\right)$. To ensure that even when the two distributions do not overlap, the distance between them can be measured. After comparing different kinds of distances and divergence property, the distance between the real distribution and the generated distribution is defined as follows:

- The *Earth-Mover* (EM) distance or Wasserstein-1:

$$W\left(P_r, P_g\right) = \inf_{\gamma\in(P_r,P_g)} E_{(x,y)\sim\gamma}[\|x - y\|], \tag{3.14}$$

where $\Pi\left(P_r, P_g\right)$ is the set of all joint distribution $\gamma(x, y)$ whose marginals are respectively P_r and P_g [28]. Simultaneously, they define a loss function for the model

as a mapping $g \mapsto \rho(P_g, P_r)$ based on the EM distance, which can be used to measure the quality of the generated distribution. The EM distance solves the problem of the *Jensen-Shannon* (JS) divergence non-convergence used in traditional GANs.

Because the infimum in Equation 3.14 is highly intractable, based on the Kantorovich-Rubinstein duality it can be rewritten as [37]:

$$\max_{D(x)\in 1-Lipschitz} E_{x\sim P_r}[D(x)] - E_{G(x)\sim P_g}[D(G(x))], \tag{3.15}$$

and solve Equation 3.4.1 where the supremum is overall the 1-Lipschitz function (K is a positive real constant).

$$\|D(x_1) - D(x_2)\| \leq K \|x_1 - x_2\|. \tag{3.16}$$

In order to enforce a Lipschitiz constraint, it is proposed to clip the weights to a fixed interval (e.g. $W = [-0.001, 0.001]^l$) after each gradient update. However, such a simple restriction on weight will lead to capacity underuse as well as exploding and vanishing gradient. Ishaan Gulrajani et al. propose a new method to clipping weight: penalize the norm of the gradient to the critical part about its input [30]. Their new objective for loss function is:

$$L = E_{\tilde{x}\sim P_g}[D(\tilde{x})] - E_{x\sim P_r}[D(x)] + \lambda E_{\hat{x}\sim P_{\hat{x}}}[(|| \bigtriangledown_{\hat{x}} D(\hat{x})||_2 - 1)^2]. \tag{3.17}$$

The specific details of the equation are described in detail in the paper [38]. The last part of Equation 3.17 (the part after λ) is a penalty on the gradient norm for random samples $\hat{x} \sim P_{\hat{x}}$. $D(x)$ is the function of discriminator and $\tilde{x}$ is the distribution generated by the generator, which is equal to $G(x)$. The specific algorithm is described as follows:

Algorithm 1 WGAN with gradient penalty

Require: The gradient penalty coefficient, λ; The number of critic iteration per generator iteration, n_{critic}; The batch size, m; Adam optimizer is chosen as the optimization algorithm. The hyperparameters of Adam, α, β_1, β_2;

Require: Initial critic parameters, w_0; Initial generator parameters, θ_0;

while θ has not converged **do**
for $t = 1, \ldots, n_{\text{critic}}$ **do**
for $i = 1, \ldots, m$ **do**
Sample real data $x \sim P_r$,latent variable $z \sim p(z)$, a random number $\epsilon \sim U[0, 1]$
$\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
$\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
$L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda \left(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1 \right)^2$
end for
$w \leftarrow \text{Adam}\left(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2\right)$
end for
Sample a batch of latent variables $\left\{\boldsymbol{z}^{(i)}\right\}_{i=1}^{m} \sim p(\boldsymbol{z})$
$\theta \leftarrow \text{Adam}\left(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w\left(G_\theta(\boldsymbol{z})\right), \theta, \alpha, \beta_1, \beta_2\right)$
end while

3.5 Learning from Incomplete Data with Generative Adversarial Networks

Based on the previous work [35] [28], Li S et al. generate a GAN based framework to learn the complex and high dimensional distribution of incomplete datasets [14]. It is further discussed in the following sections.

3.5.1 Incomplete Dataset

In a specific question, a dataset is denoted: $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{m}_i)\}_{i=1,\ldots,N}$, where $\mathbf{x} \in R^n$ is a partially observed data vector and $\mathbf{m} \in \{0, 1\}^n$ is the corresponding mask. If $m_d = 1$, x_d is observed, otherwise x_d is missing. They define a masking operator f_τ that can use a constant τ to fill in missing data. This masking operator converts incomplete data instances into vectors of the same size, where all missing items in x are replaced by the constant τ:

$$f_\tau(\mathbf{x}, \mathbf{m}) = \mathbf{x} \odot \mathbf{m} + \tau \bar{\mathbf{m}}, \tag{3.18}$$

where $\bar{\mathbf{m}}$ is the complement of set $\mathbf{m}$ and $\odot$ is element-wise multiplication.

Dataset and strategy for Generating Masks

They apply three types of missing data pattern and only one situation is described in detail below:

1. **Square available**. Only the data in a square area randomly located in the image is available, and the rest of the data is missing.

2. **Variable-size rectangular observation**. Only the data in a rectangular area that appears randomly in the image is available, and the rest of the data is lost. The area of the rectangle is random.

3. **Dropout**. Every pixel in the image is randomly lost according to the Bernoulli distribution.

The dataset used is the collection of handwritten numbers: MNIST. For each image with a size of 28×28 pixels, only a square with a size of 12×12 areas is observed, and the rest is the mask part, which is shown in Figure 3.2. Moreover, there is no dependency between the mask and the content of each image, which follows the MCAR missing data mechanisms mentioned in Section 3.1.

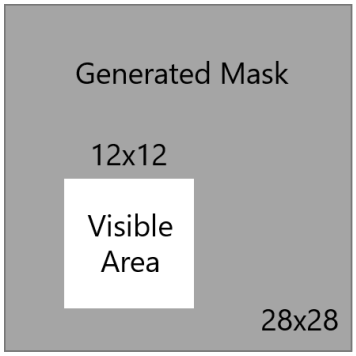


Figure 3.2: For each image with a size of 28×28 pixels, only a square with a size of 12×12 area is observed, the rest is unobservable, data which the authors describe as masked data.

3.5.2 MisGAN: a GAN for Data Imputation

Figure 3.3 shows the structure of the MisGAN. The arrows in the workflow represent the input of each part. The specific process is described below. They use generator G_m and discriminator D_m for masks as well as generator G_x and discriminator D_x for data. Random noise is input to the data generator and the mask generator to generate fake data and fake masks, respectively. Using the earlier mentioned Equation 3.18, the mask and the data are combined to create a masked data. Then

they are masked by f_τ with Equation 3.18. Similarly, real data and real masks are also masked by f_τ, then those two masked values are sent to the data discriminator. At the same time, real masks and fake masks are distinguished by the mask discriminator. As a result, compared with the traditional GAN method, MisGAN model not only learns the complete data distribution but also generates the distribution of the missing data through a mask generator. The following two loss functions for the masks and the data are defined separately. The losses follow the WGAN formulation mentioned in Section 3.4.1. It follows the WGAN-GP procedure to train discriminators with the gradient penalty:

$$\mathcal{L}_m\left(D_m, G_m\right) = E_{(\mathbf{x},\mathbf{m})\sim p_{\mathcal{D}}}\left[D_m(\mathbf{m})\right] - E_{\varepsilon\sim p_\varepsilon}\left[D_m\left(G_m(\varepsilon)\right)\right], \tag{3.19}$$

where z and ε are random noise. As a result, the optimization of the generators and discriminators are according to the following formulas:

$$\mathcal{L}_x\left(D_x, G_x, G_m\right) = E_{(\mathbf{x},\mathbf{m})\sim p_{\mathcal{D}}}\left[D_x\left(f_\tau(\mathbf{x},\mathbf{m})\right)\right] - E_{\boldsymbol{\varepsilon}\sim p_\varepsilon,\mathbf{z}\sim p_z}\left[D_x\left(f_\tau\left(G_x(\mathbf{z}), G_m(\varepsilon)\right)\right)\right]. \tag{3.20}$$

The generators and the discriminators are optimized subject to the condition that D_x and D_m conform to the restrictions, 1-Lipschitz, based on WGAN-GP mentioned in Section 3.4.1

$$\min_{G_x}\max_{D_x\in\mathcal{F}_x} \mathcal{L}_x\left(D_x, G_x, G_m\right), \tag{3.21}$$

$$\min_{G_m}\max_{D_m\in\mathcal{F}_m} \mathcal{L}_m\left(D_m, G_m\right) + \alpha\mathcal{L}_x\left(D_x, G_x, G_m\right). \tag{3.22}$$

This equation uses α equals to a small constant to force the generated masks to match the distribution of real masks as well as the generated complete samples with masks to match masked real data.

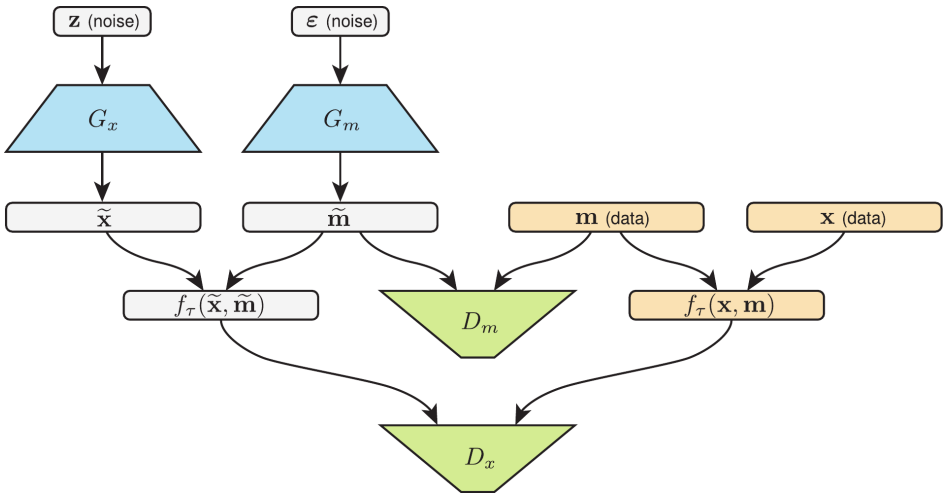


Figure 3.3: Overall structure of the MisGAN framework. The image is taken from the paper [14]

3.5.3 Missing Data Imputation

Missing data imputation is an important part when dealing with missing data. The whole framework is shown in Figure 3.5. The goal of missing data imputation is to complete the missing data according to $p(\mathbf{x}_{\text{mis}}|\mathbf{x}_{\text{obs}})$. Complete the imputation of the data through imputer G_i and the corresponding Discriminator D_i. Through the imputer G_i, the observed part of the dataset remains unchanged, while the masked part passes through $\hat{G}_i$. $\hat{G}_i$ is an imputer network that generates the imputation result. As shown in Figure 3.4, the red box is the observed part of the dataset, while the rest is generated by the imputer.

The imputer G_i is defined as follow:

$$G_i(\mathbf{x}, \mathbf{m}, \boldsymbol{\omega}) = \mathbf{x} \odot \mathbf{m} + \widehat{G}_i(\mathbf{x} \odot \mathbf{m} + \boldsymbol{\omega} \odot \overline{\mathbf{m}}) \odot \overline{\mathbf{m}}. \tag{3.23}$$

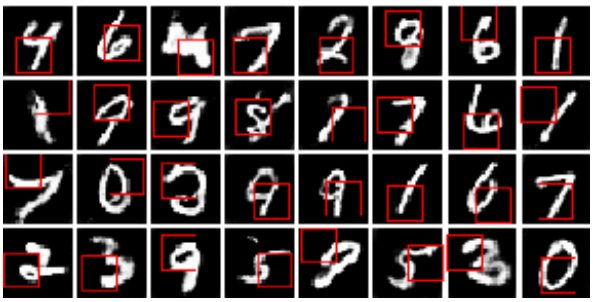


Figure 3.4: Imputation results. Inside of each red square is the observed pixels and the rest of the pixels are generated by the imputer.

The input of the imputer is the incomplete data $(\mathbf{x}, \mathbf{m})$ and a random vector ω

taken from a noise distribution. Through the observed part in x, the imputer outputs the completed sample. To train MisGAN containing the imputer, in addition to the loss functions 3.19 and 3.20 mentioned above, they defined the following loss function for the imputer:

$$\mathcal{L}_i\left(D_i, G_i, G_x\right) = E_{\mathbf{z}\sim p_z}\left[D_i\left(G_x(\mathbf{z})\right)\right] - E_{(\mathbf{x},\mathbf{m})\sim p_D, \boldsymbol{\omega}\sim p_\omega}\left[D_i\left(G_i(\mathbf{x},\mathbf{m},\boldsymbol{\omega})\right)\right]. \tag{3.24}$$

Jointly learning the data generating process and the imputer according to the following objectives:

$$\begin{aligned}
&\min_{G_i}\max_{D_i\in\mathcal{F}_i}\mathcal{L}_i\left(D_i, G_i, G_x\right),\\
&\min_{G_x}\max_{D_x\in\mathcal{F}_x}\mathcal{L}_x\left(D_x, G_x, G_m\right) + \beta\mathcal{L}_i\left(D_i, G_i, G_x\right),\\
&\min_{G_m}\max_{D_m\in\mathcal{F}_m}\mathcal{L}_m\left(D_m, G_m\right) + \alpha\mathcal{L}_x\left(D_x, G_x, G_m\right).
\end{aligned}$$

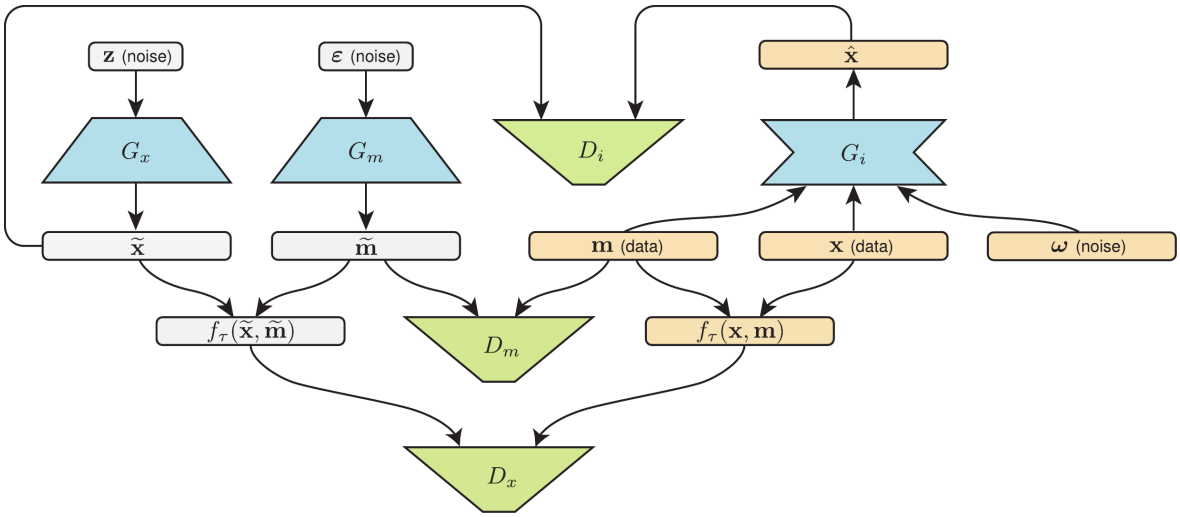


Figure 3.5: Architecture for MisGAN imputation. The image is taken from paper [14]

3.6 Evaluation Metrics

In this section, two metrics used to evaluate the performance of our models are discussed.

3.6.1 Root Mean Square Error

The Root Mean Square Error (RMSE) is usually used to compare the difference between two sequences. In this thesis, we use RMSE to calculate the gap between

the original time series and the generated time series. The definition is as follow:

$$RMSE(X, g) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(g\left(x_i\right) - y_i\right)^2}, \tag{3.25}$$

where $g(x_i)$ is the generated data and y_i is the original data. RMSE is always a non-negative value, and a value of 0 indicates a perfect fit to the data.

3.6.2 Dynamic Time Warping

Dynamic Time Warping (DTW) is a useful, powerful technique that can be applied to many different domains. Originally. it was designed to treat automatic speech recognition [39]. In time series classification, DTW is one of the algorithms for measuring similarity between two temporal sequences. These two sequences may have different speeds. It can find optimal global alignment between two time series and exploit temporal distortion between them. Figure 3.6 shows the difference between DTW and Euclidean distance. In general, DTW is a method that calculates an optimal match between two given sequences with certain restriction and rules:

- Every index from the first sequence must be matched with one or more indices from the other sequence and vice versa
- The first index from the first sequence must be matched with the first index from the other sequence (but it does not have to be its only match)
- The last index from the first sequence must be matched with the last index from the other sequence (but it does not have to be its only match)
- The mapping of the indices from the first sequence to indices from the other sequence must be monotonically increasing, and vice versa, i.e. if $j > i$ are indices from the first sequence, then there must not be two indices $l > k$ in the other sequence, such that index i is matched with index l and index j is matched with index k, and vice versa

We use the DTW method in the experiments and evaluation process to obtain the similarity of the time series of two sensors. The specific applications of DTW method will be discussed in detail in Chapter 5.

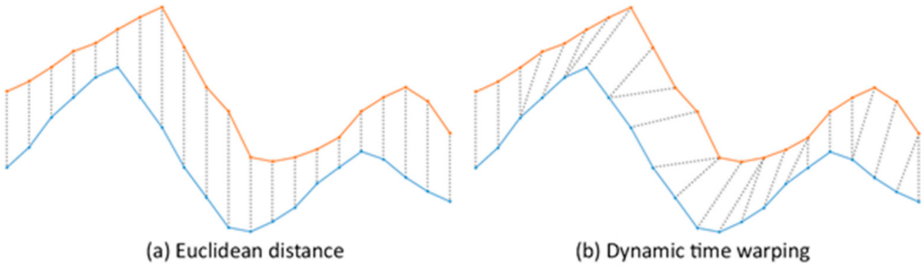


Figure 3.6: Comparison between two sequences: (a) while Euclidean distance is time-rigid, (b) the DTW is time-flexible in dealing with possible time distortion between the sequences [40].

3.7 Summary

Inspired by previous works, we intend to explore a data imputation solution for MisGAN based motion tracking. In paper [14], MisGAN is used to impute missing parts in image datasets. Similar to image data with missing parts (masks), the absence of data on sensors caused by malfunctions or failures is regarded as the ”masks” mentioned in Section 3.5.1. The complete dataset of lower body motion tracking can be generated by the MisGAN, which can be further processed and analysed in existing algorithms. In Steven Cheng-Xian Li et al.’s paper, they applied MisGAN for image data completion (e.g. MNIST dataset). In this thesis, we focus on the MisGAN model to the imputation of time series for the first time in order to obtain a complete time series datasets for human body motion tracking. Moreover, other machine learning methods are developed to do comparisons. Compared with image data, time series data is denser and changes more rapidly. We also propose metrics of accuracy and effectiveness of the algorithms which are more conducive to the time series data that we are working with. The next two chapters will now discuss the specific steps in more details, Chapter 4 discusses the specific methodologies that we applied to data imputation issues on human motion tracking data. Moreover, Chapter 5 conducts the experiments based on Chapter 4.

Chapter 4

Methodology

Most existing algorithms for motion tracking using forward kinematics based on the human skeleton model and random data missing is a challenge to these existing algorithms. The methodologies in this chapter aim to establish that the missing data from the sensor set can be estimated using supervised and unsupervised learning. These estimated values can be used to create a full information dataset, which can be further used in existing relevant algorithms. Figure 4.1 explains the whole structure of our methodologies.

In this chapter, we talk about how to apply HMMs, autoencoder and data imputation MisGAN framework mentioned in Section 3.3 to Section 3.5, to human body motion tracking dataset to impute missing data. First of all, this chapter introduces a description of the datasets used in experiments. Section 4.1 includes the characteristics and collection process of the datasets. Secondly, HMM is designed to solve the data imputation problem in Section 4.2. Then the data imputation process based on an autoencoder model is presented in Section 4.3. Finally, the MisGAN based data imputation framework is described in Section 4.4.

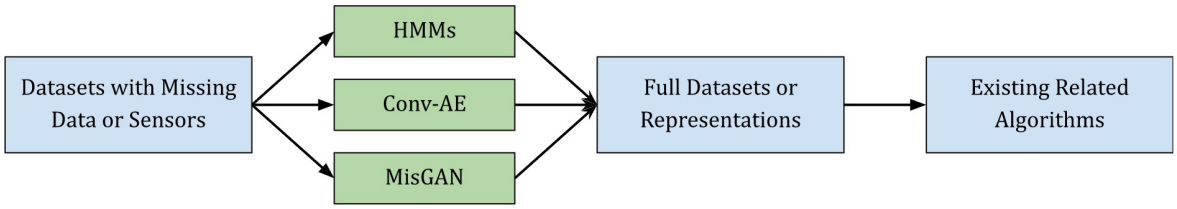


Figure 4.1: The overall process of our models.

4.1 Datasets

As Xsens MVN Link systems mentioned in Chapter 1, there is a sparse sensor setup as the original MVN solution only uses five sensors from the lower body. These five

Figure 4.2: Two persons wearing the MVN Awinda (Left) and the MVN Link (Right).

sensors are located in: pelvis, left upper leg, left lower leg, right upper leg and right lower leg. The locations of these lower body sensors are shown in the right of Figure 4.3. For each sensor, we have values from X, Y and Z axis. Therefore, we can obtain 15 values from these five sensors. The whole analysis and the proposed methodology of this chapter only focus on these five sensors of the Lower body. Moreover, to simplify the experiments from the beginning, only the data regarding the acceleration of these sensors from the lower body are used in the experiments. The acquisition frequency of the data coming from the MVN Link is of 240 Hz, which means that it contains 240 time frames per second. All the acquisitions are performed indoor.

Dataset I

A total of 8 people participated in the data collection process. For each individual, the collection steps are as follows:

1. First carry out a calibration procedure in which the tracked subjects need to stand still in the N-pose for a few seconds, walk a distance about five meters back and forth and coming back to the initial N-pose (to stand still with the arms neutral besides body) for another few seconds [42].

2. The participant was asked to walk at normal speed (depends on the participant) for 3 minutes.

The datasets of four of them are simply connected end to end as the training set,

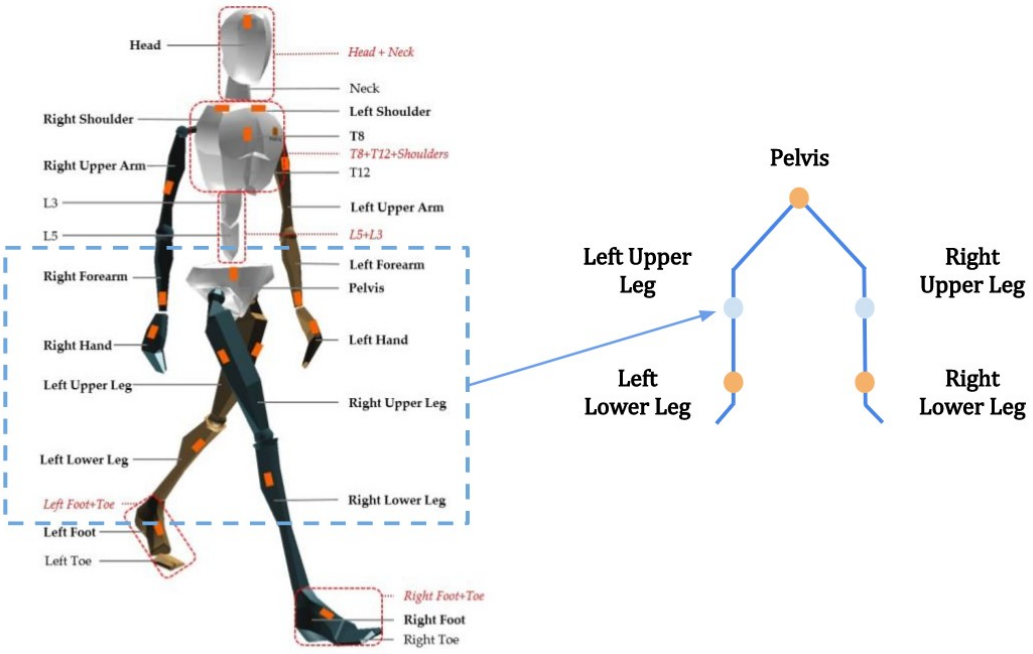


Figure 4.3: On the left there are all the sensors of Xsens MVN Link system, while on the right there are the specific five sensors from the lower body that we focus on in this thesis [41]

and the remaining data is connected end to end as the test set. The training set contains 117862 time frames and test set contains 118555 time frames.

Dataset II

A total of 8 people participated in the data collection process. For each individual, the collection steps are as follows:

1. First carry out a calibration procedure in which the tracked subjects need to stand still in the N-pose for a few seconds, walk a distance about five meters back and forth and coming back to the initial N-pose (to stand still with the arms neutral besides body) for another few seconds [42].

2. The participant was asked to jogging (the speed depends on the participant) five times back and forth, one way is about 30 meters long.

The datasets of four of them are simply connected end to end as the training set, and the remaining data are connected as the test set. The training set contains 52164 time frames and test set contains 51616 time frames.

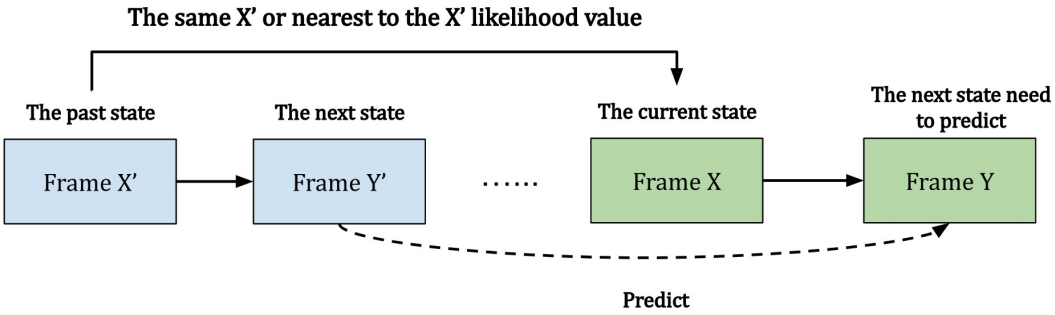


Figure 4.4: The training and prediction process of HMM.

4.2 Imputation of Human Body Motion Tracking Data based on HMMs

In Section 3.2, the working principle of HMM is presented. According to that, we design an HMM to impute those missing data with different missing rates of it. In Section 4.2.1, we briefly introduce the application of HMM.

4.2.1 Data Imputation Based on HMM

As mentioned in Section 2.2.2, we need to determine the parameters of our HMM. Using the training dataset for estimating the parameter set (A, B, π) for our HMM, then we can apply the trained HMM to predict data in future. For calculating A and B in the parameter set, we apply Forward-Backward Algorithm to train the HMM, which is a special case of the Expectation-Maximization Algorithm. For calculating π, a random number was chosen and normalized. Thus we have:

$$\sum_{i=1}^{N} \pi_i = 1.$$

With the trained HMM, likelihood values for the current time frame's data are calculated. For instance, the likelihood value for the current time frame is X. Then from the past sequence using HMM, we try to figure out the time frame that produces the same X or nearest to the X likelihood value. We then assume that the next time frame next to the current one will follow the same pattern we find above. Thus the value of the next time frame is established by adding the difference, which is calculated between the time frame we find above and the time frame next to it. Figure 4.4 shows the process mentioned above. The specific algorithm for generating parameter sets A and B is given out in [19].

4.3 Imputation of Human Body Motion Tracking Data based on autoencoder

In Section 3.3, we introduce the working principle and process of the autoencoder. In this section, we design a convolutional and deconvolutional layer based Conv-AE for the missing data imputation problem. The specific process is shown in Figure 4.5. First, we describe how to create masked data as the input of the Conv-AE in Section 4.3.1. Secondly, in Section 4.3.2, we introduce the neural network structure of the Conv-AE in detail. Finally, the procedure for data imputation on Dataset I using the Conv-AE mentioned in Section 4.3.2 is described.

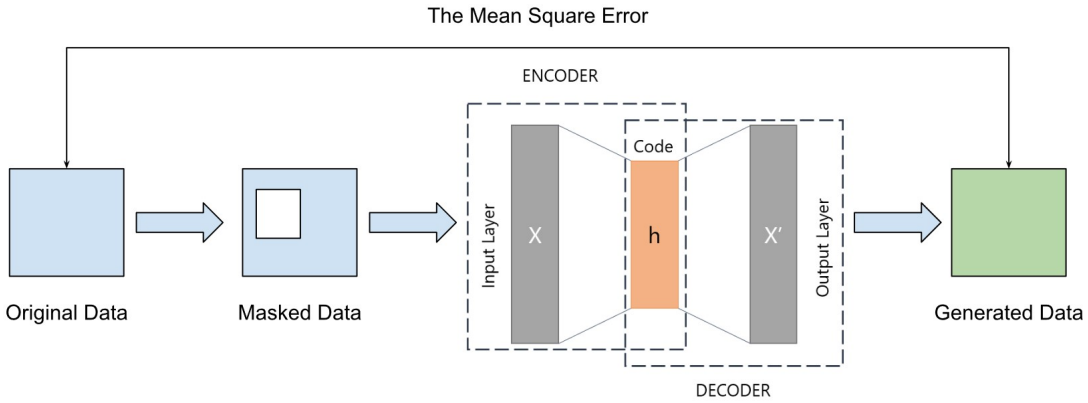


Figure 4.5: The training process of Conv-AE.

4.3.1 Masked Data Generation

In this section, we discuss how to create the input vector: $Masked\ Data$, which is shown in Figure 4.5. Firstly, as mentioned before, we only focus on the acceleration part of the datasets. For each of the five sensors, there are acceleration values for three coordinate axes: X-axis, Y-axis and Z-axis. When reading a dataset to obtain $Original\ Data$, each time frame with a length of 64 is read randomly from it. From these five sensors from the lower body, we randomly pick up one sensor to add a mask on it. When adding masks, X, Y and Z axes are all masked and the length of it is chosen from 0 to 64 (with different missing rates). The strategy of adding masks is visualized in Figure 4.6. Values from the masked data part is equal to 0, and the rest values are keeping still the original values. Finally, we get the input vector of Conv-AE: $Masked\ Data$.

With different sizes of masks, we generate parameter: $Missing\ Rate$. When

adding a mask to one of the five sensors, we calculate the missing rate as follow:

$$Missing\ Rate = \frac{Length\ of\ Mask}{Length\ of\ Time\ Frames\ Read\ from\ Dataset\ Each\ Time}$$

For instance, each time wee read a vector with the length of 64 time frames. If the mask size equals to 32, then we have the missing rate equals to 50%.

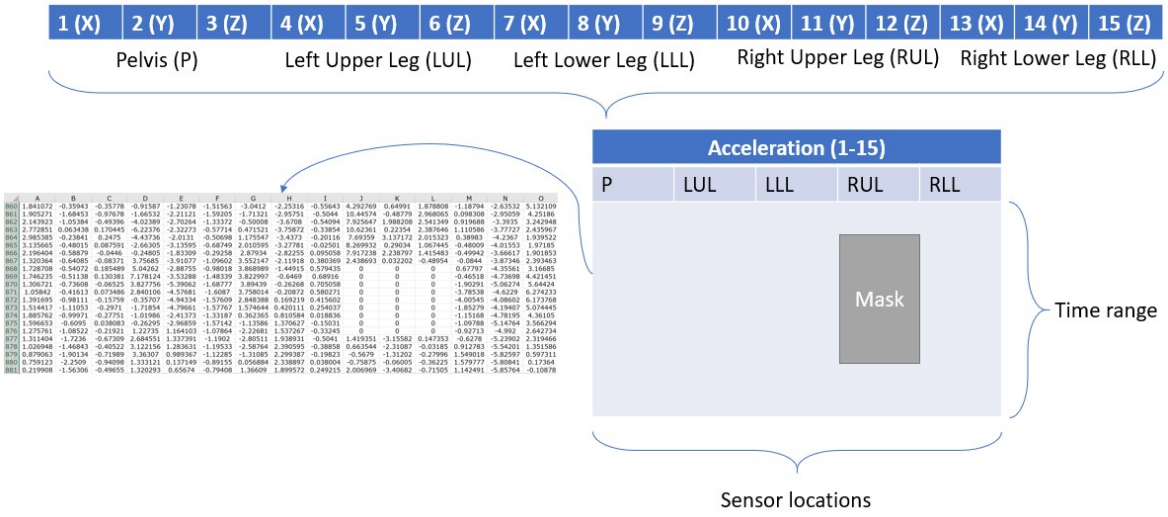


Figure 4.6: A simplified diagram of the generated mask and masked data

4.3.2 Neural Network Architecture

To create a convolutional and deconvolutional neural network based Conv-AE, we refer to the structural design of convolutional DAE [43]. For the encoder part, we generate three convolution layers with ReLU activation functions and Max Pooling 2D layers in between. For the decoder part, we apply three deconvolution layers followed by a convolution layer and with ReLUs in between [44]. The output vector of the Conv-AE has the same size as its input vector. Figure 4.7 plots the detail of the structure of Conv-AE.

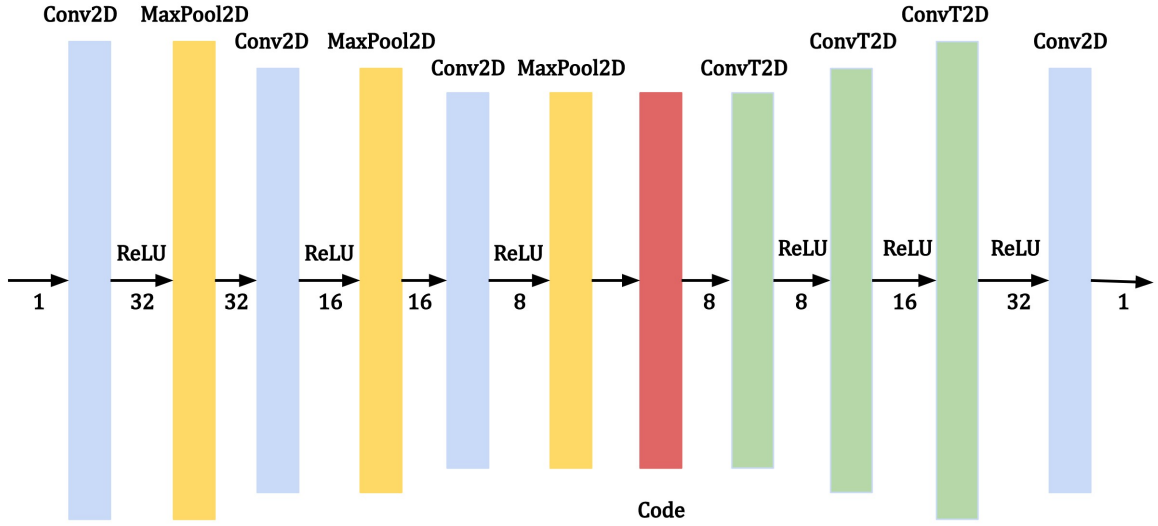


Figure 4.7: The neural network structure of DAE. [43]

4.3.3 Data Imputation Based on DAE

First, we manually add masks to the original data, where the mask represents the missing part of the data according to Section 4.3.1. We then use these masked data as the input vector of Conv-AE and get generated data after reconstruction of the Conv-AE described in Section 4.3.2. Finally, we calculate the mean square error between the original data and the generated data, which forces the Conv-AE to reconstruct data that similar to the original data. We optimize the parameters of the Conv-AE based on gradient descent via the backpropagation algorithm.

Through the above training process, the Conv-AE gains the ability to impute missing data. The Conv-AE framework trained on training datasets is used for the same type of test datasets with missing data to obtain reconstructed complete datasets. In other words, the purpose of training the model is to reduce the gap between the reconstructed data and the original data. The specific experiments and results are discussed in Section 5.2. The process are shown as follow:

1. Read the original data $x_{original\ data}$ from the dataset and the Conv-AE will receive masked data $x_{masked\ data}$ as the input vector

2. Use $x_{masked\ data}$ as the input of the DAE and the Conv-AE generates reconstructed data: $x_{generated\ data} = f_\theta(x_{masked\ data})$. Among them, $f(x)$ stands for DAE for reconstructed data and θ is the parameters in DAE, which are trained to minimize the reconstruction mean square error

3. The reconstruction mean square error (the loss function) is calculated as:

$$\mathcal{L}(x_{original\ data}, x_{generated\ data}) = \frac{1}{n}\sum_{i=1}^{n}(x_{original\ data}, x_{generated\ data})^2,$$

   n is the number of elements

4. With the loss function mentioned above, the Conv-AE is optimized by back-propagation algorithms

4.4 MisGAN based Framework for Human Body Motion Datasets

In this section, based on the MisGAN framework discussed in Section 3.5, we present how to apply the MisGAN framework for data imputation of human body motion tracking dataset. In Section 4.4.1, we describe how to desgin the input vectors of MisGAN framework. Then the architecture of neural networks used in MisGAN framework is introduced in Section 4.4.2.

4.4.1 Input Vector of MisGAN Framework Design

In this section, we introduce the input vectors (m, x, ω, etc.) of the MisGAN framework shown in Figure 4.8.

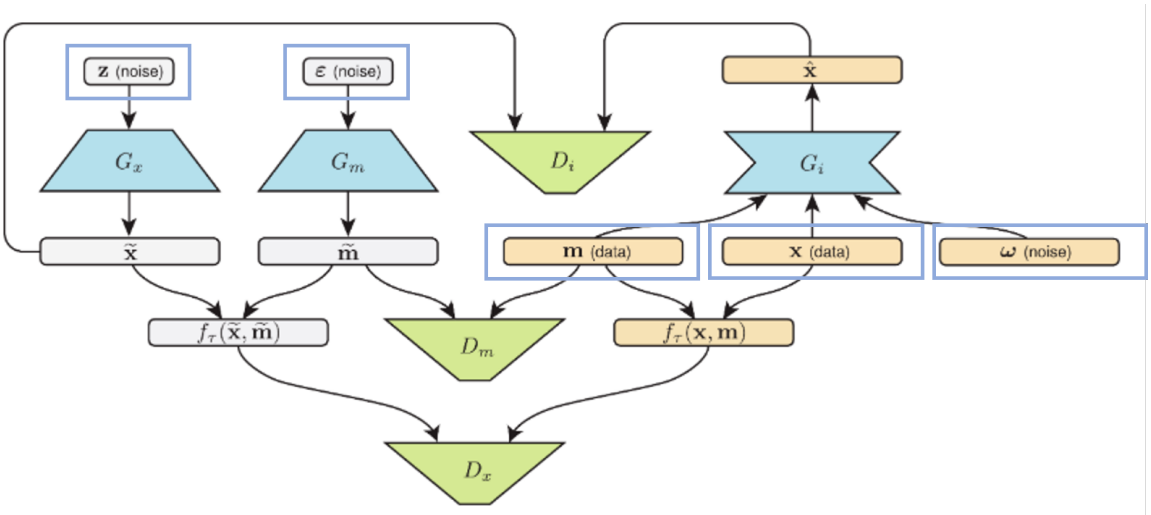


Figure 4.8: The input values of MisGAN framework. The blue squares in the figure indicate the input vectors that need to be defined.

Masked Data Generation

As discussed in Section 4.3.1, we generate the input vector: $x(data)$ similarly. We only focus on the acceleration part of the datasets mentioned in 4.1. For each of the five sensors from the lower body, there are acceleration values for three coordinate axes: X-axis, Y-axis and Z-axis. When reading a dataset, each time range with a length of 64 frames is read randomly from the training set.

Masks Generation

In this section, we describe how to generate the input vector: $m(data)$. According to Section 4.3.1, we create a mask dataset $\mathbf{m} \in \{0,1\}^n$ as the same size as the masked data in Section 5.3.2. When $m_d = 1$ means that x_d is observed, otherwise x_d is missing. The pattern of these missing data is the same as mentioned in Section 5.3.2.

Masking operator

Here we implement the mask f_τ:

$$f_\tau(\mathbf{x}, \mathbf{m}) = \mathbf{x} \odot \mathbf{m} + \tau\bar{\mathbf{m}}, \tag{4.1}$$

where $\bar{\mathbf{m}}$ is the complement of set m and $\odot$ is element-wise multiplication. As we mentioned before, an incomplete data instance can be represented as a vectors $(\mathbf{x}, \mathbf{m})$. The masking operator transforms an incomplete data into a vector of the same size with all missing data in $\mathbf{x}$ replaced by a constant value τ and τ is a constant.

Noise Generation

z and ε are two different noise taken from their own noise distribution p_z and p_ε. p_z and p_ε are standard normal distribution.

The shape of the vector p_z and p_ε are:

$$[batch\,size, in\_features = 128]$$

The noise ω is taken from standard normal distribution p_ω. The shape of the vector is the same as $m\ (data)$ and $x\ (data)$

4.4.2 Neural Network Architecture

In this thesis, we refer to the structural design of deep neural networks by Steven Cheng-Xian Li et al. to create generators, discriminators and imputer [45]. With these well-defined neural network units, we generate the MisGAN based framework shown in Figure 3.5.

Generator

For each generator, we generate a linear layer followed by three deconvolution layers with ReLUs in between shown in Figure 4.9. In particular, we apply a sigmoid activation function defined in Equation 4.2 for the output of the mask generator G_m in order to keep the output of the G_m closer to zero or one.

$$\sigma_\lambda(x) = \frac{1}{1 + exp(-x/\lambda)}, \tag{4.2}$$

where $0 < \lambda < 1$.

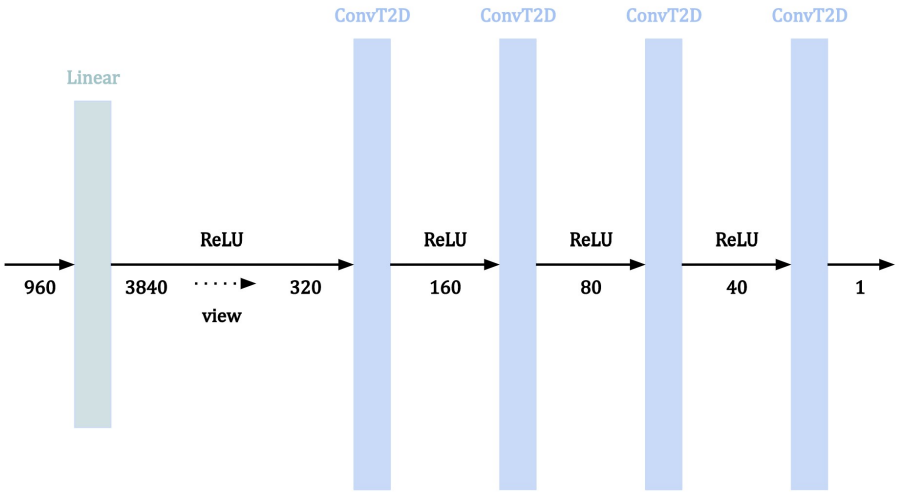


Figure 4.9: The neural network structure of the generator G. The numbers in the figure represent the number of input or output units. Besides, use "view" to reshape the tensors in the neural network to adjust the input shape.

Discriminator

We implement the discriminator (or referred to as the critic in Wasserstein GANs [28]) with three convolutional layers followed by a linear layer for both D_x discriminator for data and D_m discriminator for masks in Figure 4.10.

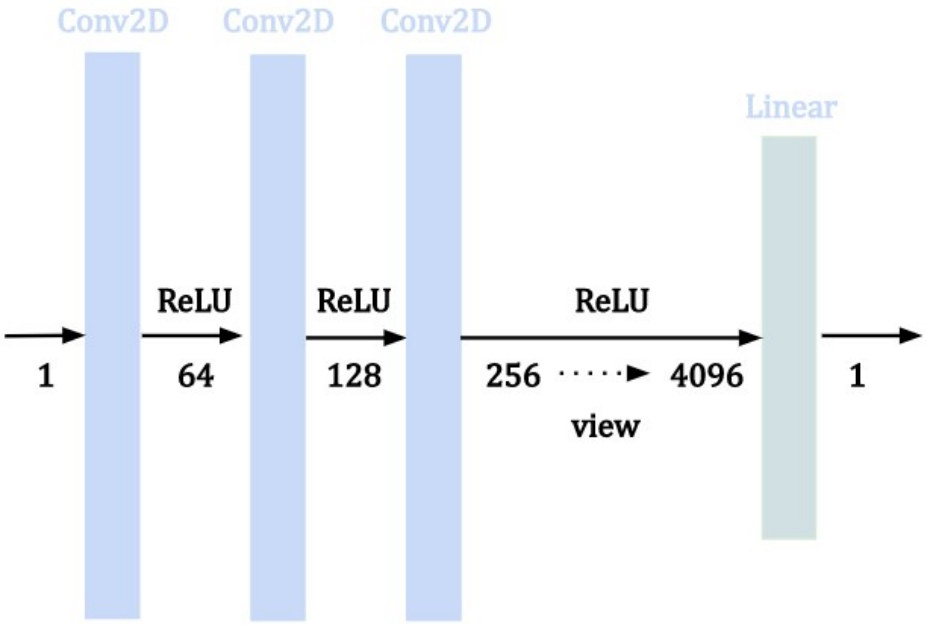


Figure 4.10: The neural network structure of the discriminator D. The numbers in the figure represent the number of input or output units. In addition, use "view" to reshape the tensors in the neural network to adjust the input shape.

Imputer

We generate G_i as a three-layer fully-connected network with ReLUs activation function in between in Figure 4.11.

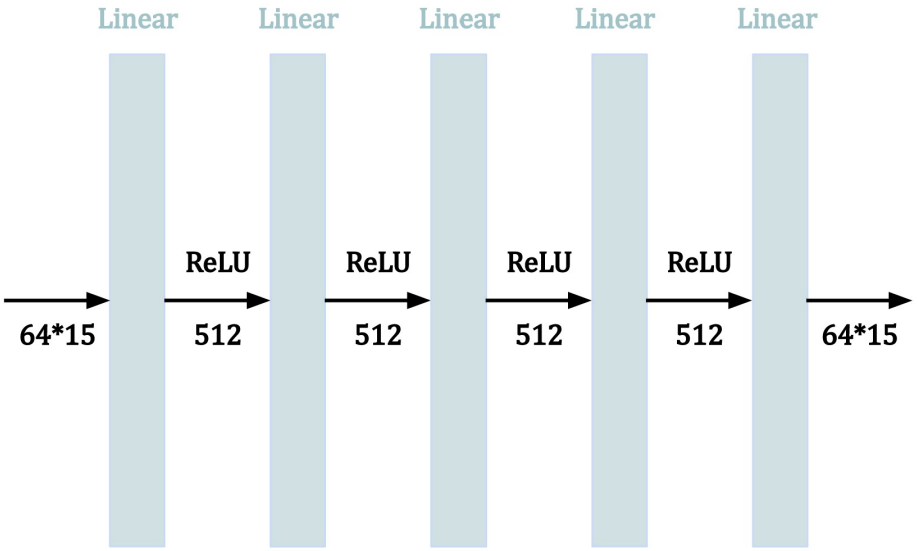


Figure 4.11: The neural network structure of the imputer G_i. The numbers in the figure represent the number of input or output units.

4.5 Imputation of Human Body Motion Tracking Data based on MisGAN

In this section, first of all, based on Section 4.4.2, we use the G_x, G_m and f_τ to generate $f_\tau(\tilde{\mathbf{x}}, \tilde{\mathbf{m}})$ indistinguishable from the masked real incomplete data $f_\tau(\mathbf{x}, \mathbf{m})$ with discriminator D_x. The specific experimental methods and results are discussed in Section 5.3.2.

Secondly, according to Section 4.4.2, we use the imputer G_i and the corresponding discriminator D_i to impute the missing part of our dataset. The generated data and the original data are plotted to show the intuitive visual comparison between the expected and estimated data windows in Section 5.3.1.

4.5.1 MisGAN for Human Body Motion Tracking Datasets

In this section, based on the framework shown in Figure 3.3, we apply data generator G_x, mask discriminator G_m as well as corresponding data discriminator D_x, mask discriminator D_m to generate $f_\tau(\tilde{\mathbf{x}}, \tilde{\mathbf{m}})$ similar to the masked real incomplete data $f_\tau(\mathbf{x}, \mathbf{m})$.

The loss functions for this process are defined in Equation 3.20 for data discriminator and 3.19 for mask discriminator. Then we can optimize the generators and discriminators according to Equation 3.21 and 3.22, which meet 1-Lipschitz conditions according to paper [28]. At the same time, we follow the usual practice of GAN based framework training, which is to alternate between optimization discriminators several times and optimization generators several times. The specific experiments process and results are shown in Chapter 5.

4.5.2 Missing Data Imputation for Human Body Motion Datasets

In this section, we indicate how to impute missing data by equipping imputer G_i and corresponding discriminator D_i.

The imputer is a function of the incomplete data (x, m) and a random vector ω that taken from a noise distribution p_ω. The formula is shown as follow:

$$G_i(\mathbf{x}, \mathbf{m}, \boldsymbol{\omega}) = \mathbf{x} \odot \mathbf{m} + \widehat{G}_i(\mathbf{x} \odot \mathbf{m} + \boldsymbol{\omega} \odot \overline{\mathbf{m}}) \odot \overline{\mathbf{m}} \tag{4.3}$$

Same as in paper [14], we apply loss function 3.20, 3.19 and 3.24. As a result, we can jointly learn the data generating process and imputer G_i according to Equation 3.5.3.

As a result, through the function of this function, G_i will generate new imputed data in the masked part of the original dataset with missing data. When $m = 1$, the

above equation is: $G_i(\mathbf{x}, \mathbf{m}, \boldsymbol{\omega}) = \mathbf{x}$. Otherwise, when $m = 0$, the above equation is: $G_i(\mathbf{x}, \mathbf{m}, \boldsymbol{\omega}) = \widehat{G}_i(\boldsymbol{\omega})$.

To validate the proposed methods of data imputation in the sparse sensor setup, we experiment on the dataset described in Section 4.1. We attempt to impute the missing acceleration values of the sensor using MisGAN framework.

The loss functions mentioned in Section 4.5.1 and Section 4.5.2 meet 1-Lipschitz conditions according to paper [14]. At the same time, we follow the usual practice of GAN based framework training, which is to alternate between optimization discriminators several times and optimization generators several times.

4.6 Summary

In this chapter, we first discuss two datasets we use in this thesis. Then we develop three machine learning algorithm based models to deal with data imputation issues on human motion tracking datasets, which have different characteristics and application fields. In the next chapter, we describe the specific experiment procedures, parameters and model structure, etc., and then show the experiment results.

Chapter 5

Experiments and Results

In this chapter, we conduct the experiments inspired by the algorithms discussed in Chapter 4. These experiments aim at generating the missing data or sensors of human body motion tracking using HMM, Conv-AE as well as MisGAN based framework. These missing values in human motion tracking datasets can be caused due to a sensor missing for the entire recording, or in the event of hardware or software failure.

This chapter describes the experiments and reports the results obtained. The experiments in this chapter verify the feasibility of applying HMM, Conv-AE and MisGAN base framework to deal with missing data imputation for human body motion tracking dataset. Our models and corresponding evaluation experiments are implemented using PyTorch [46] from Python, and the experiment processes are run on a GPU: GeForce GTX 1080.

5.1 Data Imputation Based on HMM

In this section, we evaluate our HMM model on the training set of Dataset I introduced in Section 4.1. In these experiments, we use $hmmlearn$ library in Python, which is an open source library to train the model, calculate the likelihood of the observation and parameters of the HMM model. In each prediction, we use 500 time frames of data. We keep aside the last several observations for testing and used the rest of them to train the model. For the number of states, we calculate the negative log-likelihood of the training set of each model and choose the model has the lowest value with the restriction of BIC value. The optimal number of states is determined by the algorithm in $hmmlearn$ library.

5.1.1 Results and Discussion

Similarly as the aforementioned section. To evaluate how good the HMM performed, we apply the two metrics discussed in Section 3.6.2 and 3.6.1 to test our model with different lengths of mask: 8, 16 and 32. In the experiments, we assume that at a certain moment the signal from the chosen sensor becomes 0, that is, the signal disappears. The length of the time sequence of lost signal is the length of the mask. As a generative model of supervised learning, HMMs need to generate a model and its parameters through previous sequences. We train our HMMs with a time sequence of length 500 before the signal disappears and then do the prediction for a time sequence with the different size of masks: 8, 16 and 32.

Especially, we only compare the gap between the generated data and the original data of the masked part by performing calculations on the whole training set (In One Epoch) of Dataset I and take the average. At the same time, we also compare the HMM results with some simple methods: Zero Imputation and Mean Imputation. These two data imputation methods are single imputation mentioned in Section 2.1.2.

By observing the results in Table 5.1 and 5.2, with the metric of DTW, our HMM performs better than two simple single imputation methods. However, with the evaluation of RSME, there is no obvious improvement effect by applying HMMs.

Imputation Strategy \ Size of the Mask	8	16	32
Zero Imputation	21.664546	45.815017	88.421334
Mean Imputation	21.871443	45.443609	88.624361
HMM	**20.068047**	**28.466772**	**72.235297**

Table 5.1: DTW is applied to calculate the gap between original data and imputed data.

Imputation Strategy \ Size of the Mask	8	16	32
Zero Imputation	1.573452	2.354871	2.587613
Mean Imputation	1.579386	2.358139	2.584171
HMM	**4.444647**	**4.439613**	**4.419325**

Table 5.2: RMSE is applied to calculate the gap between original data and imputed data.

5.1.2 Visualization of Data Imputation on HMM

This section plots the comparison between the predicted data and the original data. Similarly, we use 500 time frames to train our HMM model and then apply the HMM we obtain to do a prediction for the next 240 time frames. Because the sampling frequency of the MVN Link system is $240Hz$, we take 240 time frames for our visualization, which equivalent to simulation for one second. Figure 5.1 shows the result for our visualization for 240 time frames with HMM of the sensor on right upper leg (X-axis). The visualization results from other two axes are shown in Appendix A.1.

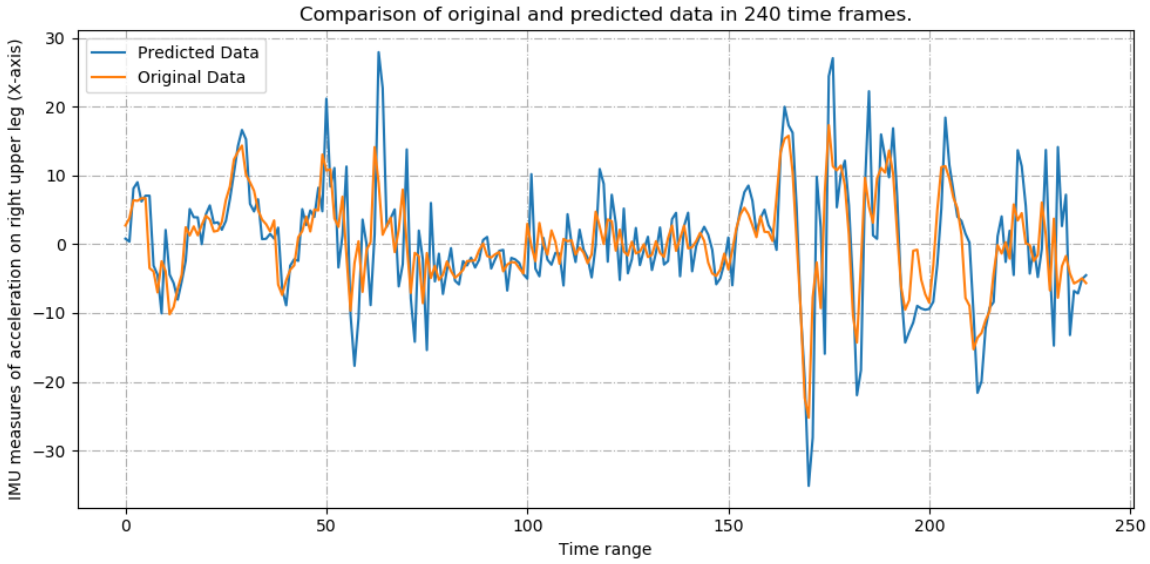


Figure 5.1: Comparison of the original and the imputed IMU measures of the acceleration part from HMM on the right upper leg (X-axis) for 240 time frames (Dataset I).

5.2 Data Imputation Based on DAE

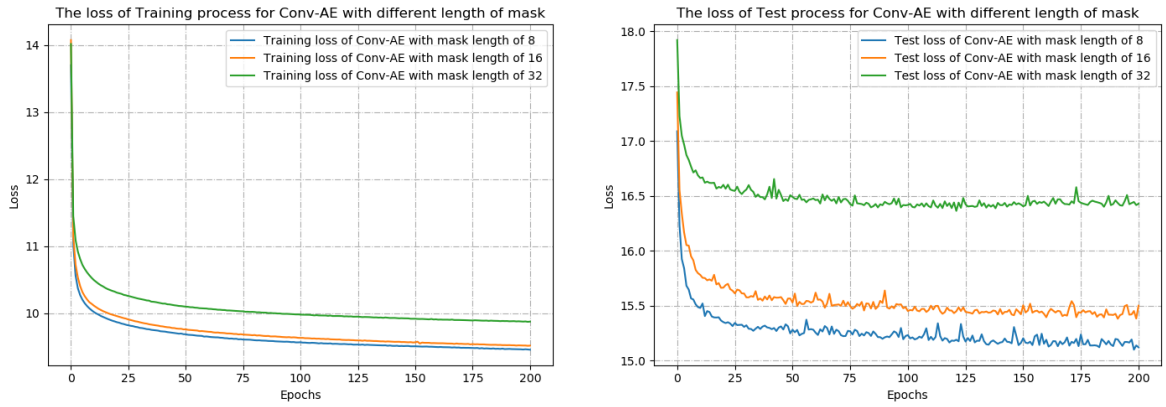
In this section, we evaluate our Conv-AE model on Dataset I. This dataset includes a training set and a test set, each of which consists of 4 people walking at a normal speed. Details are introduced in Section 4.1. The structure we use is shown in Figure 4.5.

Moreover, some parameters of the Conv-AE model are decided:

- Learning rate, $\lambda = 1e - 3$
- Batch size, $batch\_size = 128$
- Epochs of training and test, $epoch = 200$

- Parameters in Adam optimizer: $betas = (0.9, 0.999)$
- Missing rate, $Missing\ Rate = [0.125, 0.25, 0.5]$
- The shape of the input vector, [batch size, number of channels, height in pixels, width in pixels]$\rightarrow [128, 1, 64, 15]$

For this experiment, we train the Conv-AE on the training set of Dataset I for 200 epochs and the entire training set is traversed in each epoch. In each batch, we input a vector with a length of 64 time frames. At the same time, the model is tested on the test set of Dataset I. During the training and test process, we plot the mean square error between the original data and the generated data with different lengths of mask: 8, 16 and 32 (Missing Rate=[0.125, 0.25, 0.5]) in Figure 5.2.



(a) Loss during the training process with different size of mask

(b) Loss during the test process with different size of mask

Figure 5.2: Training and test loss of Conv-AE with different size of mask.

5.2.1 Results

In order to evaluate the performance of our Conv-AE performed, we apply metrics discussed in Section 3.6.2 and 3.6.1 to test our model with different lengths of mask: 8, 16 and 32. (Missing Rate=[0.125, 0.25, 0.5]). Especially, we only compare the gap between the generated data and the original data of the masked part. In the entire training set, we perform calculations in the 200^{th} epoch for a vector with a length of 64 time frames and take the average of them.

In the reported Table 5.3 and 5.4, it is possible to observe that with the DTW metric, our Conv-AE decreases the difference between the original complete data and the data generated by our Conv-AE compare to two single imputation methods.

Imputation Strategy \ Size of the Mask	8	16	32
Zero Imputation	21.963547	45.622097	88.922368
Mean Imputation	21.681453	45.749508	88.412032
Conv-AE	**20.255444**	**44.935510**	**75.137959**

Table 5.3: DTW is applied to calculate the gap between original data and imputed data.

Imputation Strategy \ Size of the Mask	8	16	32
Zero Imputation	1.578851	2.353271	2.588315
Mean Imputation	1.575381	2.357739	2.587146
Conv-AE	**1.594706**	**2.437528**	**2.632676**

Table 5.4: RMSE is applied to calculate the gap between original data and imputed data.

5.2.2 Visualization of Data Imputation on Conv-AE

In this Section, we do visualization for comparison between the data generated by Conv-AE and the original data. We apply the model we obtain after 200 epochs in the aforementioned section to do this visualization with a longer sequence for 240 time frames in Figure 5.3. The result is for the sensor on right upper leg (X-axis). The results for the other two axes for the same sensor are plotted in Appendix A.2.

5.3 Data Impuation Based on MisGAN

In this Section, we develop our MisGAN model for data imputation on Dataset I introduced in Section 4.1. Firstly, we evaluate the possibility of applying the MisGAN to our human motion tracking datasets. Secondly, we do some preliminary experiments focus on the structure discussed in Section 3.5.2 to visualize how good the generated results are. Finally, we conduct experiments on data imputation focus on the structure in Section 3.5.3 of MisGAN with doing comparisons with simple data imputation methods and the two machine learning based data imputation algorithms in the aforementioned sections.

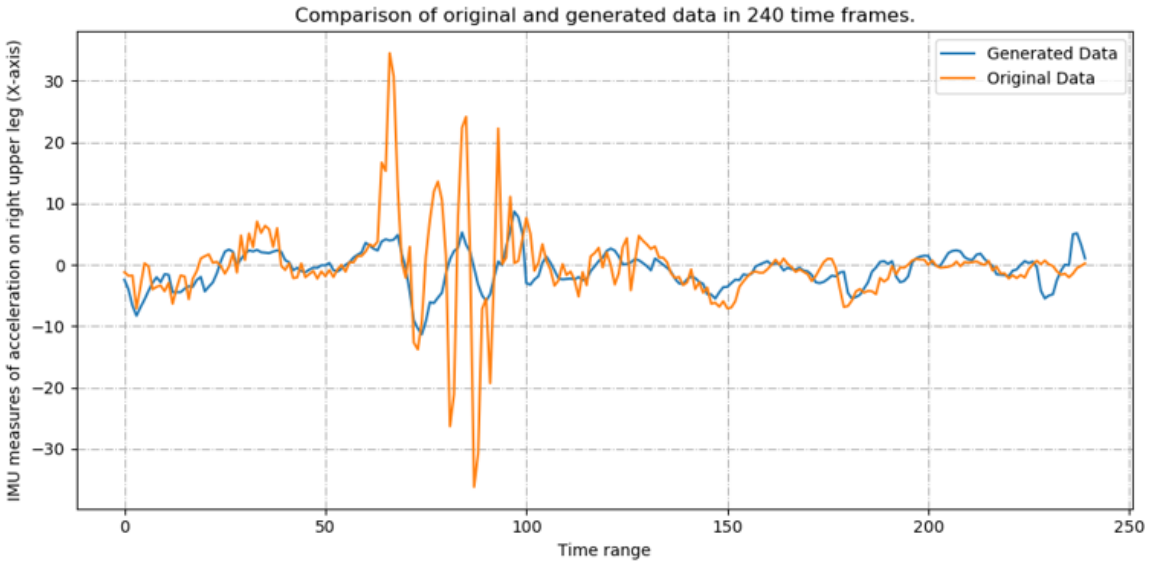


Figure 5.3: Comparison of the original and the generated IMU measures of the acceleration part from Conv-AE on the right upper leg (X-axis) for 240 time frames (Dataset I).

5.3.1 Possibility of Applying MisGAN

In the previous research, most of the GAN based frameworks were used for image processing and generation. GAN based frameworks are rarely applied for processing time series datasets. For evaluating the possibility of applying MisGAN to the human body motion tracking dataset, we first visualize the loss functions for discriminators and imputer defined in Section 3.5. Obviously, the loss functions reflect the gap between the distribution of generated data and the distribution of original data. We take absolute values of the output values of all those loss functions, so the closer the value of the loss is to 0, the closer the generated data is to the original data. The value of the loss function reflects that the generators is constantly learning how to generate more accurate data during the training process. In this section, we conduct experiments based on the framework mentioned in Section 3.5.3. In this experiment, we use the training set from Dataset I (4.1) to train the MisGAN framework shown in Figure 3.5 for 400 epochs. In each batch, we input a vector with the length of 64 time frames and the mask size of this experiment is equal to 16 time frames.

Some parameters of the MisGAN model are decided as follows:

- Learning rate: $\lambda = 1e - 4$
- Batch size, $batch\_size = 128$
- Epochs of training, $epoch = 400$

- Parameters in Adam optimizer, $\beta = (0.5, 0.9)$

- Missing rate, $Missing\ Rate = 0.25$

- The shape of the input vector, [batch size, number of channels, height in pixels, width in pixels]$\rightarrow [128, 1, 64, 15]$

Results

We can draw conclusions by observing the results: the loss of discriminator for data, masks and imputer show an overall downward trend in 400 epochs. The loss values of discriminators for data generator, mask generator and imputer are reducing during the training process, which means that the MisGAN is learning how to generate the data with the same distribution as the original dataset. In conclusion, it is possible for us to apply MisGAN framework to human motion body tracking datasets. Figure 5.4 to 5.6 show the specific changes in the discriminator's loss function within the training process.

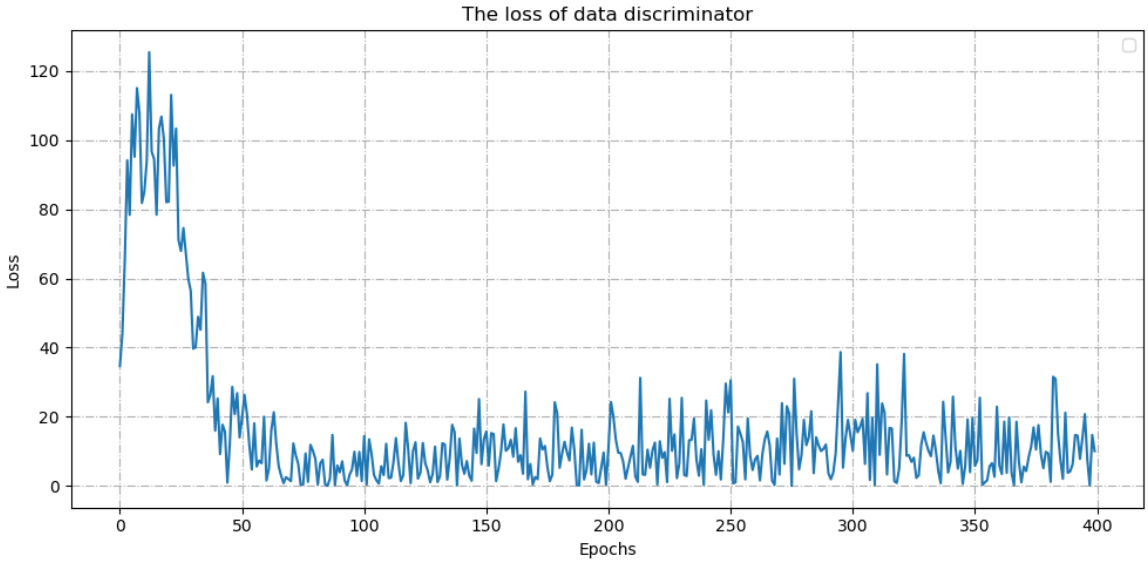


Figure 5.4: Loss value of D_x in 400 epochs

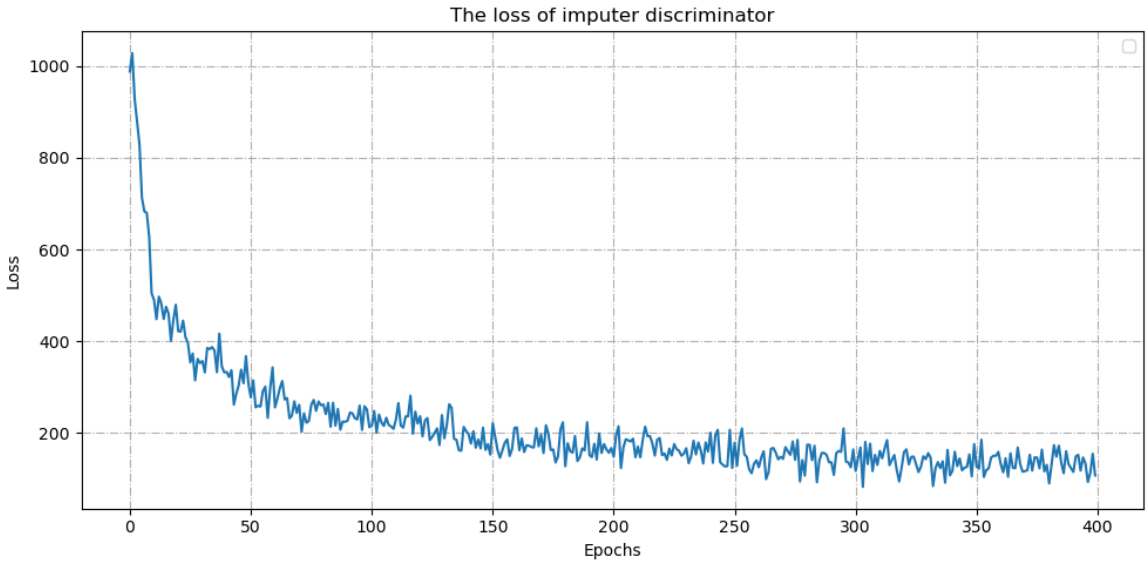


Figure 5.5: Loss value of D_i in 400 epochs

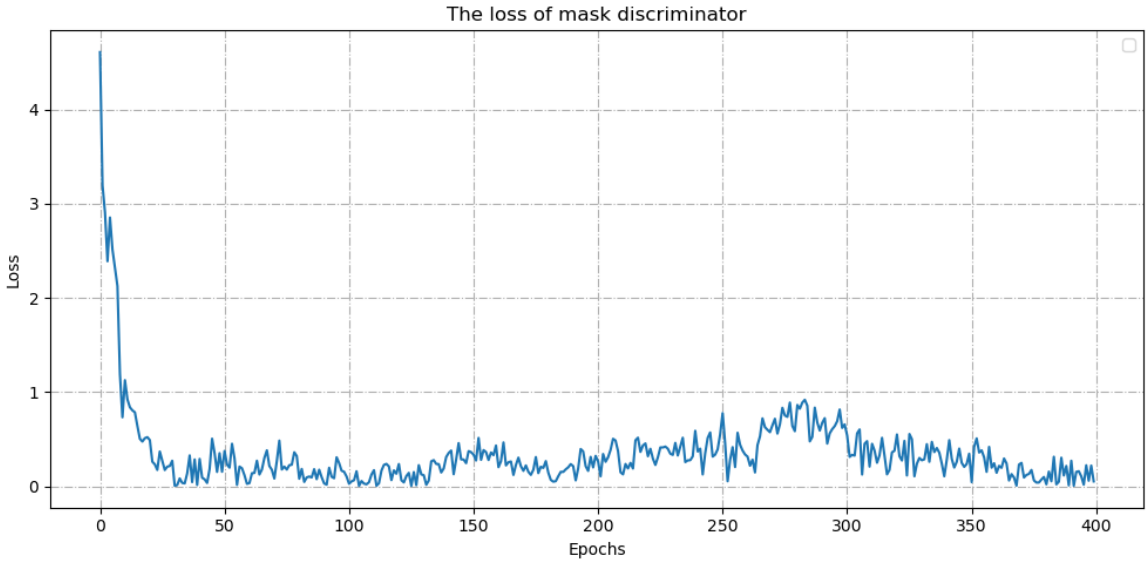


Figure 5.6: Loss value of D_m in 400 epochs

5.3.2 Masked Data Generation

In this section, we carry out experiments based on the framework shown in Figure 3.3. We use generator G_x for masked data, G_m for masks and f_τ to generate $f_\tau(\mathbf{x}, \mathbf{m})$. Then we apply discriminator D_x and let $f_\tau(\mathbf{x}, \mathbf{m})$ indistinguishable from the masked real incomplete data $f_\tau(\mathbf{x}, \mathbf{m})$.

To validate the proposed methods of data generation in the framework in Section 4.4, we conducted experiments on the dataset described in Section 4.1. The Mis-

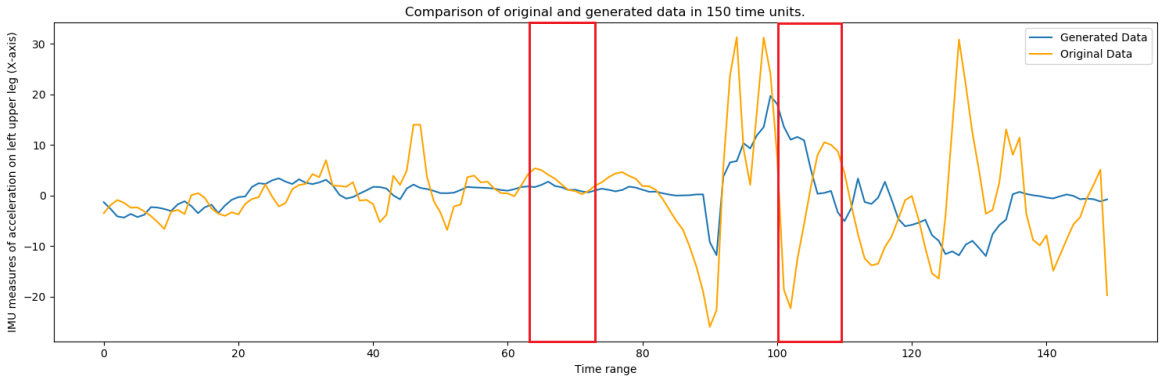


Figure 5.7: Comparison of the original and the generated IMU measures of the acceleration part on the left upper leg (X-axis). Time range from 64 to 72 and 100-108 are masked.

GAN architecture learns to recreate the acceleration of a given sensor after observing the full sequence of the sensor recordings having masked values. Effectively, the sensor masks in the full acceleration sequence are filled by the MisGAN. We train the discriminator and generator for 250 epochs with Adam optimizer with a learning rate 10^{-4}. For training WGAN with gradient penalty, we use all the default hyperparameters applied in paper [28]. In this section, we will now describe the results of the experiments in some detail.

Preliminary Experiments

The entire time series is continuously read in a time series of 45 lengths. In other words, each time range with a length of 45 is read one by one from the whole dataset. We generate two datasets as the input, namely the Mask and the Masked Data and set them as the input of the MisGAN framework: $\mathbf{m}$ and $\mathbf{x}$. At the same time, we use random noise as input to data and mask generator. Therefore, the inputs $\mathbf{m}$ and $\mathbf{x}$ of the MisGAN framework will be two dimensional: $time\ range \times IMU\ Measures = 45 \times 45$.

We plot the comparison of the original and generated data for 150 time units on the left upper leg and pelvis in Figure 5.7. More images of the experimental results are in Appendix A.3.

Conclusion

With these experiments conducted in Section 5.3.2, by observing the results of reading the entire dataset continuously, it is found that the generated can better fit the original data, but it may have certain specificity and may not be applicable to time series data in different situations. In these experiments, we only visually visualize

the results of data completion. In the following experiments, we focus on the data imputation part of MisGAN and develop metrics to analyse the quality of correction achieved using these methods.

5.3.3 Data Imputation with MisGAN

Based on the methodology discussed in Section 4.5, we carry out the experiment for data imputation on MisGAN. We conduct the experiments on Dataset I and Dataset II. This dataset contains a training and a test set, each of which consists of 4 people walking at a normal speed. Details are introduced in Section 4.1.

Some parameters of the MisGAN model are decided as follows:

- Learning rate: $\lambda = 1e - 4$
- Batch size, $batch\_size = 128$
- Epochs of training, $epoch = 400$
- Parameters in Adam optimizer, $\beta = (0.5, 0.9)$
- Missing rate, $Missing\ Rate = [0.125, 0.25, 0.5]$
- The shape of the input vector, [batch size, number of channels, height in pixels, width in pixels]$\rightarrow [128, 1, 64, 15]$

For this experiment, we train the MisGAN model for 400 epochs with different learning rate (Missing Rate=[0.125, 0.25, 0.5]) and the entire training set is traversed in each epoch.

Results and Discussion

In order to evaluate how our model performs, the two metrics introduced in Section 3.6.1 and 3.6.2 are applied to our training set in Dataset I with different learning rate (Missing Rate=[0.125, 0.25, 0.5]). We only compare the gap using the two metrics between the generated data and the original data of the masked part. In the entire training set, we perform calculations in the 400^{th} epoch for inputting vectors with a length of 64 time frames and take the average of them. Similarly, as before, we do comparisons among MisGAN model and two single data imputation algorithms. The results on Dataset I and II are shown in Table 5.5 to 5.8.

Imputation Strategy \ Size of the Mask	8	16	32
Zero Imputation	28.718901	51.394067	112.414015
Mean Imputation	28.727416	51.963509	112.764129
MisGAN	**15.424699**	**25.756295**	**55.985976**

Table 5.5: DTW is applied to calculate the gap between original data and imputed data (Dataset I).

Imputation Strategy \ Size of the Mask	8	16	32
Zero Imputation	1.671638	2.957502	2.144072
Mean Imputation	1.672439	2.956034	2.154157
MisGAN	**1.225829**	**1.707168**	**2.477405**

Table 5.6: RMSE is applied to calculate the gap between original data and imputed data (Dataset I).

Imputation Strategy \ Size of the Mask	8	16	32
Zero Imputation	79.546674	130.331768	319.435240
Mean Imputation	79.426123	131.421742	319.441254
MisGAN	**73.675379**	**72.131524**	**199.315319**

Table 5.7: DTW is applied to calculate the gap between original data and imputed data (Dataset II).

Imputation Strategy \ Size of the Mask	8	16	32
Zero Imputation	4.391675	6.816055	9.941666
Mean Imputation	4.564742	6.932234	9.674574
MisGAN	**4.535847**	**4.182040**	**5.657422**

Table 5.8: RMSE is applied to calculate the gap between original data and imputed data (Dataset II).

Moreover, here we also output the comparison results calculated by DTW between the three machine learning based algorithms in order to highlight the performance of MisGAN on Dataset I.

Imputation Strategy \ Size of the Mask	8	16	32
HMM	20.068047	28.466772	72.235297
Conv-AE	20.255444	44.935510	75.137959
MisGAN	**15.424699**	**25.756295**	**55.985976**

Table 5.9: DTW is applied to calculate the gap between original data and imputed data (Dataset I).

Visualization of Data Imputation on MisGAN

This section plots the comparison between the data generated by the imputer and the original datasets. Different from the visualization mentioned before, the working principle of imputer is different from the machine learning based model mentioned above. The imputer only imputes the masked part of the data, and the rest remains unchanged. We apply the imputer of MisGAN model of 400^{th} epoch in Section 4.4 to do data generation for 240 time frames in Figure 5.8 and 5.9 with mask size of 32, which is the imputed results from sensor on right upper leg (X-axis) on Dataset I and II. Results from other two axes are plotted in Appendix A.4. In the following figures, the parts of the blue line are the values generated by the imputer. By observing the blue curves generated by the imputer, we can figure out that they reflect the changing trend of the original data. Compared with the original data, generating data is more gentle.

Figure 5.8: Comparison of the original and the generated IMU measures of the acceleration part from Imputer in MisGAN on the right upper leg (X-axis) for 240 time frames (Dataset I).

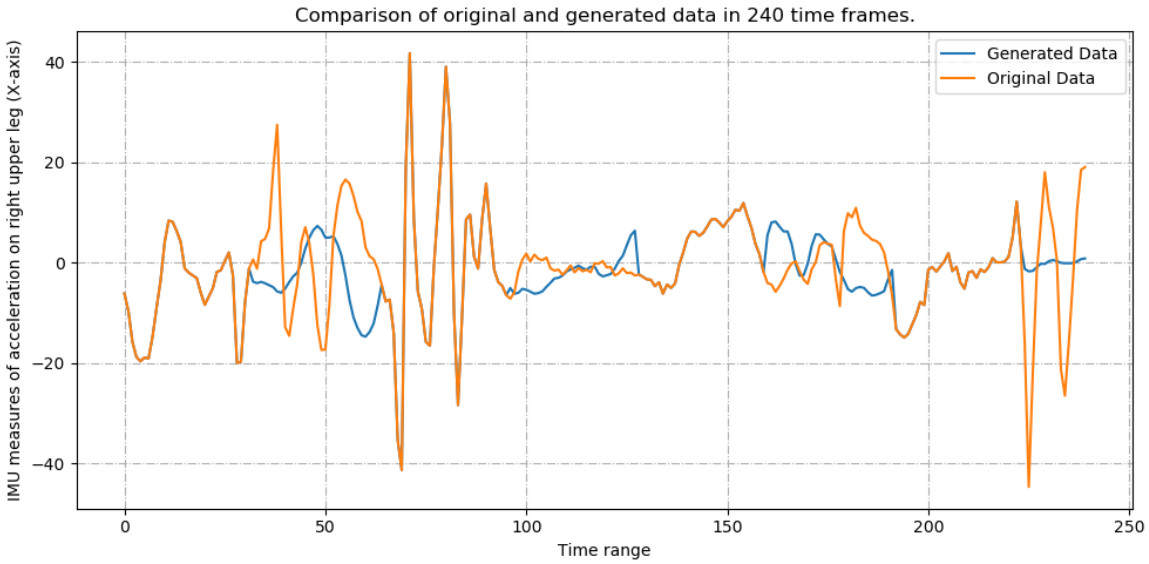


Figure 5.9: Comparison of the original and the generated IMU measures of the acceleration part from Imputer in MisGAN on the right upper leg (X-axis) for 240 time frames (Dataset II).

5.4 Discussion

In Section 5.1, we conduct experiment based on HMM. From the visualization in Figure 4.4, in some points when the original data changes rapidly, the data generated by the HMM changes more drastically and the fluctuation range is greater than the original data, which is the reason for the larger value with RSME. When the data does not change significantly, the HMM fits the original data better. However, when HMM predicts the missing data of a certain sensor during a period of time, it needs data from the sequence before the sensor fails to train the model. The trained model can then be adopted to impute the missing data. Therefore, it is clear that HMM is not suitable to establish a sparse sensors setup, because the data from the reduced sensor is missing for the whole sequence and the HMM cannot be trained.

Section 5.2 explains the application of autoencoders. As an unsupervised learning method, autoencoder can learn the hidden information of the dataset. The new features learned by the autoencoder can be fed into the supervised learning model, thus reducing the need for 'new and labelled data'. However, compared with HMM, Conv-AE cannot reflect such changes in the generated data when the data undergoes large and rapid changes.

In Section 5.3.3, we explore the possibility of exploiting MisGAN to our human motion tracking dataset. The results show that it is feasible to apply MisGAN to human motion tracking data, and the effect is greater than other methods for compari-

son. MisGAN can be trained using an unlabeled human motion tracking dataset as they can learn the internal representations of the dataset. It allows a deep learning model to capture the distribution of the input training dataset and generate accurate results. Thus without any information from the previous sequence or other sensors, MisGAN can generate new data with the same distribution as the original data.

In general, through the experiments in this chapter, the results meet our expectations. We apply two metrics: RMSE and DTW, to evaluate the experiment results. In order to highlight the effects of the three machine learning algorithm based data imputation methods on our human motion datasets, we also conduct experiments on two single imputation approaches. We set different data missing rates to carry out experiments on our models and get the results. With the application of DTW, the three machine learning methods are more exceptional than the two single completion methods with different missing rates. Table 5.1, 5.3 and 5.5 indicate the specific results on Dataset I and II. Comparing these three machine learning models in Table 5.9 on Dataset I, MisGAN obtained the best results. In the case of different missing rates, compared to Zero Imputation and Mean Imputation, the DTW value is reduced to about half of the original. However, with the application of RMSE, the results of machine learning approaches do not show a significant improvement compare to the two single imputation methods. RMSE represents the mean value of the sum of squared errors of the corresponding points between the predicted data and the original data. Since the mean value of our human motion tracking datasets is close to 0, using zeros or the mean value of the available part of the dataset to impute the missing values, the calculated RMSE value will be smaller. But obviously using these two single data imputation methods lacks some practical significance, because they cannot fit the characteristics of time series data changes well. In this chapter, experiment results prove that MisGAN can be applied to generate missing parts in the human motion dataset, and the effect is greater than HMM and Conv-AE. Thus MisGAN can be used to impute time series data.

Chapter 6

Conclusions and Future Work

In this chapter, we first explain how we achieve the Research Goals we listed in Chapter 1 by briefly concluding our findings and contributions. Finally, Future Work for this thesis is discussed.

6.1 Conclusions

- (RG 1) *To study the previously used methods for data imputation and analyze their characteristics and limitations. Explain why these traditional data imputation methods are not suitable for human body motion data and the reason why choosing neural networks or unsupervised learning based methods.*

  For RG 1, we have studied several traditional data imputation methods. In general, these methods are not suitable for human motion tracking datasets for the following reasons: 1. High computational complexity on large datasets. 2. Not suitable for time series data imputation 3. The accuracy of the generated data is low, etc. These inspire us to find more suitable methods for data imputation for human motion tracking data. Machine learning algorithms provide solutions for solving sparse sensor setup and missing data imputation issues in various domains. Machine learning algorithms give us solutions to assist in missing data with high uncertainty estimation scenarios. Their capabilities indicate that they can generate data with more complex distributions.

- (RG 2) *To investigate machine learning algorithms that have been employed in the problems of data imputation. Discusses the possibility of applying these methods to human body motion data imputation area.*

  For RG 2, we apply HMM, Conv-AE, and MisGAN based framework to investigate data imputation issues. As a kind of unsupervised learning, autoencoder can be used to discover latent patterns in datasets and extract information

from it. HMM is a type of generative model that predicts future data through past sequences and can be used as an imputation to missing data. MisGAN framework can be used to learn the complex and high dimensional distribution of incomplete datasets in image processing and generation.

- (RG 3) *To explore and develop algorithms selected in Research Goal 2, which help to compensate for the missing data in the human body motion datasets or to minimize the need for 'new and labelled data' for human body motion tracking datasets.*

  For RG3, we design models based on HMMs, autoencoders and MisGAN framework and conduct specific experiments on them. We design the structures and experiments based on different parameters. Under the condition of applying DTW as the evaluation metrics, HMM, autoencoder and MisGAN have achieved better results than the two traditional single imputation algorithms. Among them, MisGAN achieved the best results. But when using RMSE to do the evaluation, the three machine learning algorithms are not significantly improved compared to the two traditional methods.

- (RG 4) *To develop metrics to report the quality of correction achieved using the aforementioned methods.*

  For reporting the quality of the aforementioned methods, we develop two metrics: DTW and RMSE to evaluate the results we obtain from experiments. As a result, we discover that DTW can better reflect the difference between the time series.

This thesis uses the MisGAN framework to impute human motion tracking datasets with different missing rates. This is the first time that MisGAN framework has been applied to time series datasets. At the same time, we also generate models based on HMMs and autoencoders, compare these results and evaluate them. Using MisGAN to impute the missing data, the processed complete datasets can be used in existing algorithms and learning models, which makes it possible for us to use a reduced sensors set up or improve the robustness of the motion tracking system.

While experimenting with three machine learning algorithms, we also developed two single imputation methods to highlight the superiority of the three machine learning algorithms. Under the evaluation of DTW, the effects of the three machine learning methods are better than the two single imputation methods. Taking the experimental result with a missing rate of 50% as an example, HMM reduces the DTW value by 18.3% compared to Zero Imputation. Compared with Mean Imputation, HMM reduces the DTW value by 18.5%. For Conv-AE, DTW is reduced by 15.5% and 15.0%, respectively. In particular, MisGAN reduced the DTW value by 50.2%

and 50.4% separately. However, with RMSE, the effects of the three machine learning algorithms and the two traditional methods are not significantly different. Thus, we can conclude that DTW is more suitable to evaluate time series data (human motion tracking data) than RMSE.

This thesis can be applied as solutions for data imputation issues of human motion tracking data. Through those aforementioned data imputation algorithms, we are able to obtain complete datasets, which can be applied to existing algorithms for human motion tracking models. At the same time, having the ability to generate complete datasets can also be applied to reduce the number of sensors required in motion capture in order to reduce the cost and complexity of usage.

6.2 Future Work

We only conduct experiments on relatively small datasets, which may lead to the lack of generalization ability and robustness of our machine learning based models. In future experiments, we will apply larger scale datasets with multiple types of human movement, and divide them into training sets and test sets.

The tuning of the parameters and the optimal structure of the models are not be explored thoroughly. A systematic and rigorous process should be done by tuning the parameters and structures until reaching the optimal results. For example, different values of the learning rate, numbers of layers of neural networks, etc. Especially, for the structure of our neural networks, we need to explore the structure that more suitable to human motion tracking data, which can take the sensors' locations and features into consideration. With these modifications, we are able to improve the performance of our existing models.

This thesis only studies the data imputation of the acceleration part of our human motion tracking datasets. In addition to acceleration data, there is also orientation data in our human motion tracking datasets. In the future, research will be conducted on different characteristics of orientation data and to deal with data imputation issues on it.

This research does not take the connection among different sensors into consideration when imputing missing data. When there are some missing values in a certain sensor, the remaining sensors can still work normally. The information contained in the data collected from available sensors can be used for data imputation of missing values. For instance, we can calculate the gap between the time series from the sensor with missing data and time series from other available sensors. Thus the available sensor values can be input into our imputation models to improve the accuracy of the generated data.

This thesis develops innovative machine learning based solutions for data imputation issues in the human body motion tracking area. The complete datasets after being imputed by our models will be used in the existing algorithm, and the effect of applying the complete original dataset will be compared to observe the effect of data imputation.

Bibliography

[1] A. Safonova, J. K. Hodgins, and N. S. Pollard, “Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces,” in *SIGGRAPH 2004*, 2004.

[2] T. Sanger, “Human arm movements described by a low-dimensional superposition of principal components,” *The Journal of neuroscience : the official journal of the Society for Neuroscience*, vol. 20, pp. 1066–72, 03 2000.

[3] M. Soley-Bori, “Dealing with missing data: Key assumptions and methods for applied analysis,” 2013.

[4] M. Schepers, M. Giuberti, and G. Bellusci, “Xsens mvn: Consistent tracking of human motion using inertial sensing,” 03 2018.

[5] J.-F. Cai, E. J. Candès, and Z. Shen, “A singular value thresholding algorithm for matrix completion,” *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010. [Online]. Available: https://doi.org/10.1137/080738970

[6] R. Mazumder, T. Hastie, and R. Tibshirani, “Spectral regularization algorithms for learning large incomplete matrices,” *J. Mach. Learn. Res.*, vol. 11, p. 2287–2322, Aug. 2010.

[7] H. Kang, “The prevention and handling of the missing data,” *Korean journal of anesthesiology*, vol. 64, 2013.

[8] J. M. Jerez, I. Molina, P. J. García-Laencina, E. Alba, N. Ribelles, M. Martín, and L. Franco, “Missing data imputation using statistical and machine learning methods in a real breast cancer problem,” *Artificial Intelligence in Medicine*, vol. 50, no. 2, pp. 105 – 115, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0933365710000679

[9] H. Barlow, “Unsupervised learning,” *Neural Computation*, vol. 1, no. 3, pp. 295–311, 1989. [Online]. Available: https://doi.org/10.1162/neco.1989.1.3.295

[10] J. Brownlee, “A tour of machine learning algorithms,” 2019, https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/ Accessed July 20, 2020.

[11] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” *CoRR*, vol. abs/1311.2901, 2013. [Online]. Available: http://arxiv.org/abs/1311.2901

[12] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F. Wang, “Generative adversarial networks: introduction and outlook,” *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 588–598, 2017.

[13] R. Fu, J. Chen, S. Zeng, Y. Zhuang, and A. Sudjianto, “Time series simulation by conditional generative adversarial net,” 2019.

[14] S. C. Li, B. Jiang, and B. M. Marlin, “Misgan: Learning from incomplete data with generative adversarial networks,” *CoRR*, vol. abs/1902.09599, 2019. [Online]. Available: http://arxiv.org/abs/1902.09599

[15] J. Bennett and S. Lanning, “The netflix prize,” in *Proceedings of the KDD Cup Workshop 2007*. New York: ACM, Aug. 2007, pp. 3–6. [Online]. Available: http://www.cs.uic.edu/~liub/KDD-cup-2007/NetflixPrize-description.pdf

[16] P. D. Allison, “Handling missing data by maximum likelihood,” *SAS Global Forum 2012 Statistics and Data Analysis*, 2012. [Online]. Available: http://www.statisticalhorizons.com/wp-content/uploads/MissingDataByML.pdf

[17] Y. Obadia, “The use of knn for missing values,” 2017, https://towardsdatascience.com/the-use-of-knn-for-missing-values-cf33d935c637 Accessed May 3, 2020.

[18] B. L and S. A, “Nearest neighbor imputation algorithms: a critical evaluation,” *BMC medical informatics and decision making*, vol. 74, 2016. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4959387/

[19] D. Jurafsky and J. H. Martin, *Speech and Language Processing*. Stanford University, 2020.

[20] M. Hassan and B. Nath, “Stockmarket forecasting using hidden markov model: A new approach.” vol. 2005, 10 2005, pp. 192 – 196.

[21] N. M. Nguyen, “Stock price prediction using hidden markov model,” 2016.

[22] P. Pandey, “Deep generative models,” 2018, https://medium.com/@prakashpandey9/deep-generative-models-e0f149995b7c Accessed May 4, 2020.

[23] J. T. McCoy, S. Kroon, and L. Auret, “Variational autoencoders for missing data imputation with application to a simulated milling circuit,” *IFAC-PapersOnLine*, vol. 51, no. 21, pp. 141 – 146, 2018, 5th IFAC Workshop on Mining, Mineral and Metal Processing MMM 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896318320949

[24] V. Fortuin, D. Baranchuk, G. Rátsch, and S. Mandt, “Gp-vae: Deep probabilistic time series imputation,” *arXiv: Machine Learning*, 2019.

[25] H. minn Lu, G. Perrone, and J. Unpingco, “Multiple imputation with denoising autoencoder using metamorphic truth and imputation feedback,” *ArXiv*, vol. abs/2002.08338, 2020.

[26] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” 2014.

[27] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” *CoRR*, vol. abs/1710.10196, 2017. [Online]. Available: http://arxiv.org/abs/1710.10196

[28] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein gan,” 2017.

[29] C. T. F. L. Hitchcock, Y. Rubner and L. J. Guibas, “The earth mover’s distance,” http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/RUBNER/emd.htm Accessed September 20, 2020.

[30] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein gans,” *CoRR*, vol. abs/1704.00028, 2017. [Online]. Available: http://arxiv.org/abs/1704.00028

[31] Y. Huang, M. Kaufmann, E. Aksan, M. J. Black, O. Hilliges, and G. Pons-Moll, “Deep inertial poser: Learning to reconstruct human pose from sparse inertial measurements in real time,” *CoRR*, vol. abs/1810.04703, 2018. [Online]. Available: http://arxiv.org/abs/1810.04703

[32] S. Li, Y. Zhou, H. Zhu, W. Xie, Y. Zhao, and X. Liu, “Bidirectional recurrent autoencoder for 3d skeleton motion data refinement,” *Computers & Graphics*, vol. 81, pp. 92 – 103, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0097849319300330

[33] Z. Wang, J. Chai, and S. Xia, "Combining recurrent neural networks and adversarial training for human motion synthesis and control," *CoRR*, vol. abs/1806.08666, 2018. [Online]. Available: http://arxiv.org/abs/1806.08666

[34] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," 2015.

[35] J. Yoon, J. Jordon, and M. van der Schaar, "Gain: Missing data imputation using generative adversarial nets," 2018.

[36] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *ArXiv*, vol. abs/1701.04862, 2017.

[37] C. Villani, *Optimal Transport*. Grundlehren der mathematischen Wissenschaften, 2009.

[38] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *CoRR*, vol. abs/1606.03498, 2016. [Online]. Available: http://arxiv.org/abs/1606.03498

[39] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[40] A. G. K. M. Csillik O, Belgiu M, "Object-based time-constrained dynamic time warping classification of crops using sentinel-2," *Remote Sensing*, vol. 11(10):1257, 2019.

[41] A. Karatsidis, G. Bellusci, M. Schepers, M. de Zee, M. Andersen, and P. Veltink, "Estimation of ground reaction forces and moments during gait using only inertial motion capture," *Sensors*, vol. 17, p. 75, 01 2017.

[42] Xsens, "Mvn user manual," 2020, https://www.xsens.com/hubfs/Downloads/Manuals/MVN_User_Manual.pdf Accessed August 2, 2020.

[43] L. Barbosa, "Convolution autoencoder - pytorch," 2019, https://www.kaggle.com/ljlbarbosa/convolution-autoencoder-pytorch Accessed August 2, 2020.

[44] T. Contributors, "Torch.nn," 2019, https://pytorch.org/docs/stable/nn.html Accessed August 2, 2020.

[45] S. C. Li, B. Jiang, and B. M. Marlin, "Misgan: Learning from incomplete data with gans," 2019, https://github.com/steveli/misgan/blob/master/misgan.ipynb Accessed July 13, 2020.

[46] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8026–8037. [Online]. Available: http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Appendix A

Appendix

A.1 Visualization Results of HMM

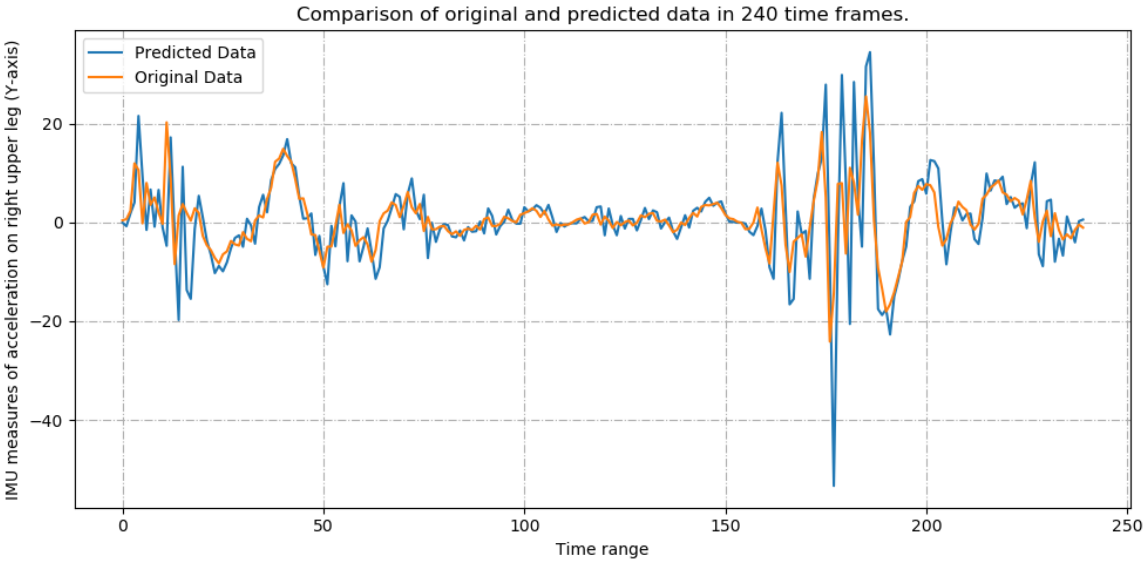


Figure A.1: Comparison of the original and the imputed IMU measures of the acceleration part from HMM on the right upper leg (Y-axis) for 240 time frames (Dataset I).

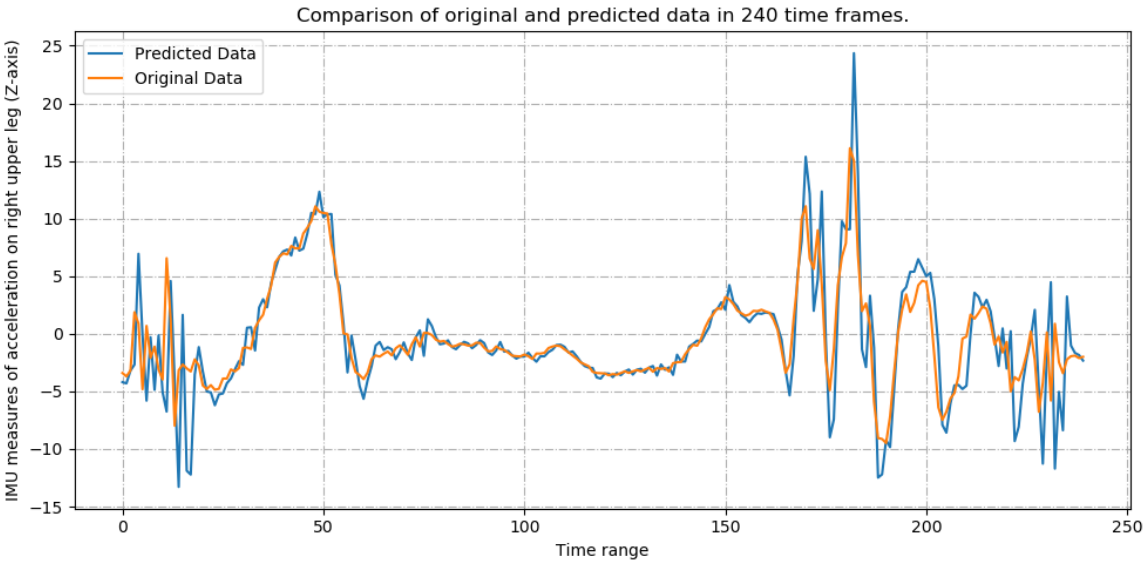


Figure A.2: Comparison of the original and the imputed IMU measures of the acceleration part from HMM on the right upper leg (Z-axis) for 240 time frames (Dataset I).

A.2 Visualization Results of Conv-AE

Figure A.3: Comparison of the original and the generated IMU measures of the acceleration part from Conv-AE on the right upper leg (Y-axis) for 240 time frames (Dataset I).

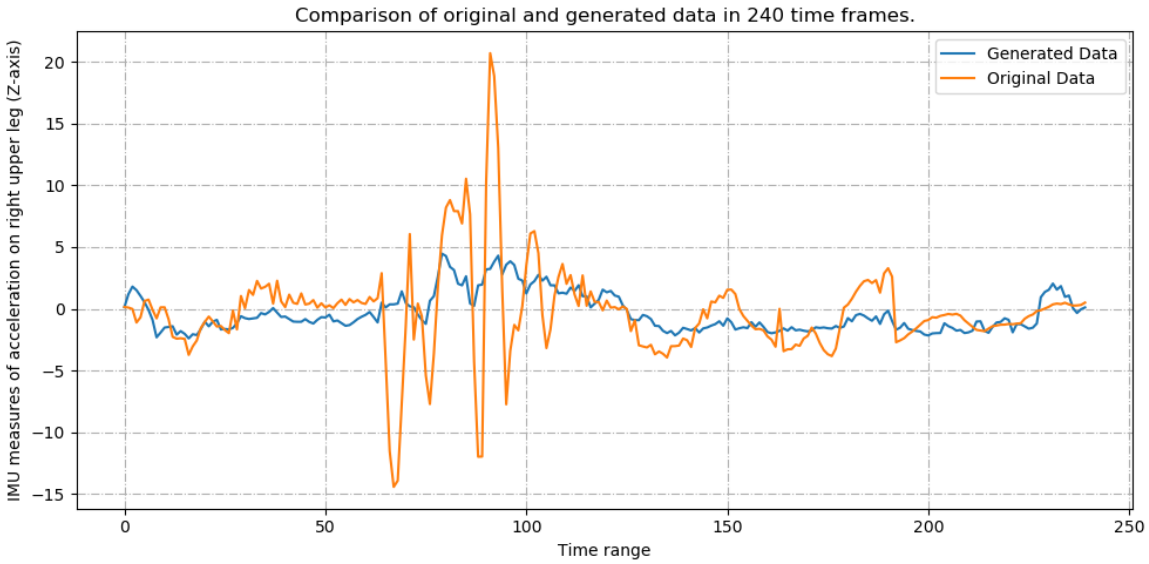


Figure A.4: Comparison of the original and the generated IMU measures of the acceleration part from Conv-AE on the right upper leg (Z-axis) for 240 time frames (Dataset I).

A.3 Data Generation

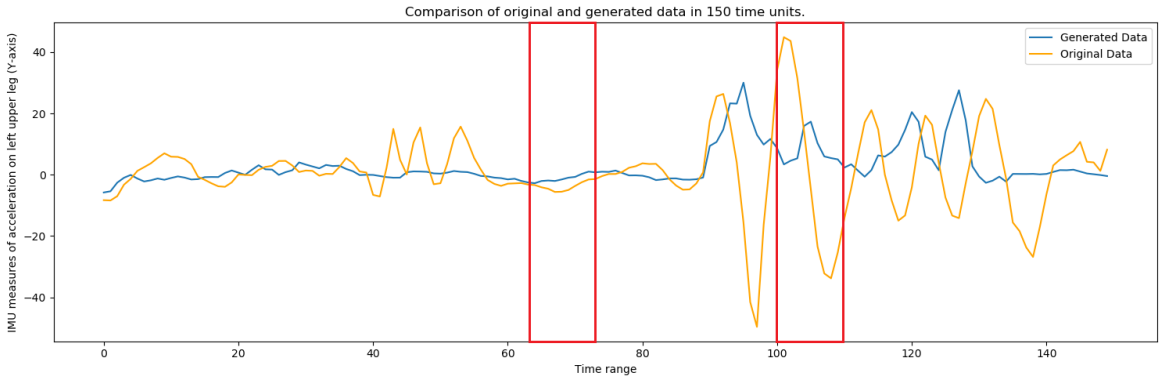


Figure A.5: Comparison of the original and the generated IMU measures of the acceleration part on the left upper leg (Y-axis) Time range from 64 to 72 and 100-108 are masked.

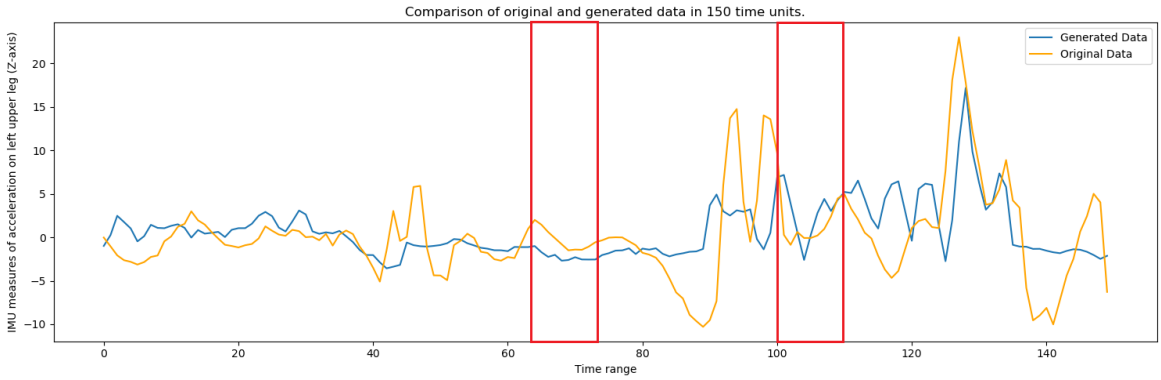


Figure A.6: Comparison of the original and the generated IMU measures of the acceleration part on the left upper leg (Z-axis) Time range from 64 to 72 and 100-108 are masked.

A.4 Visualization Results of MisGAN

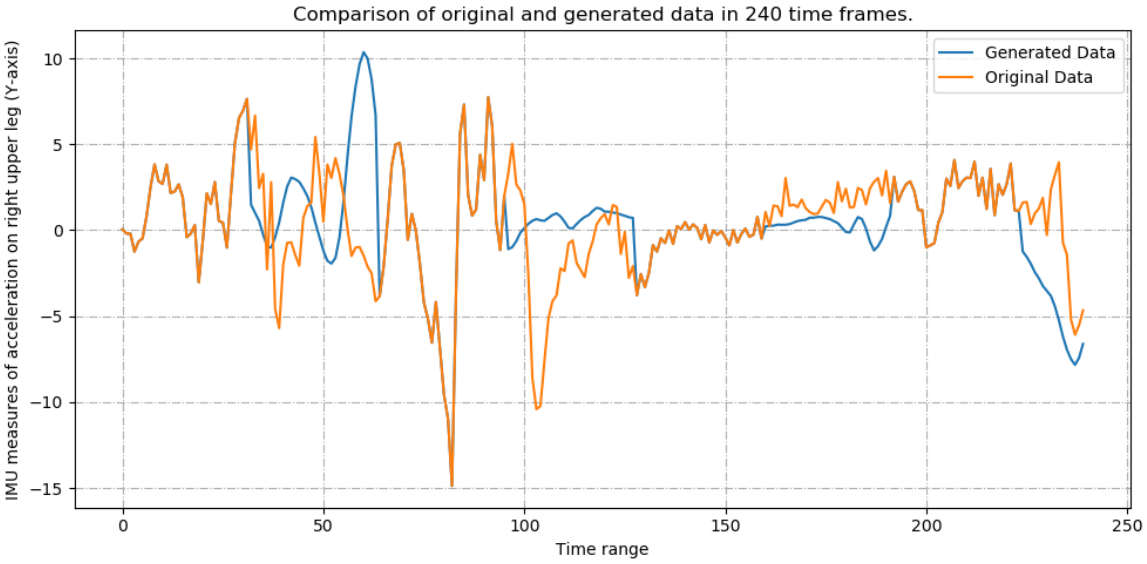


Figure A.7: Comparison of the original and the generated IMU measures of the acceleration part from Imputer in MisGAN on the right upper leg (Y-axis) for 240 time frames (Dataset I).

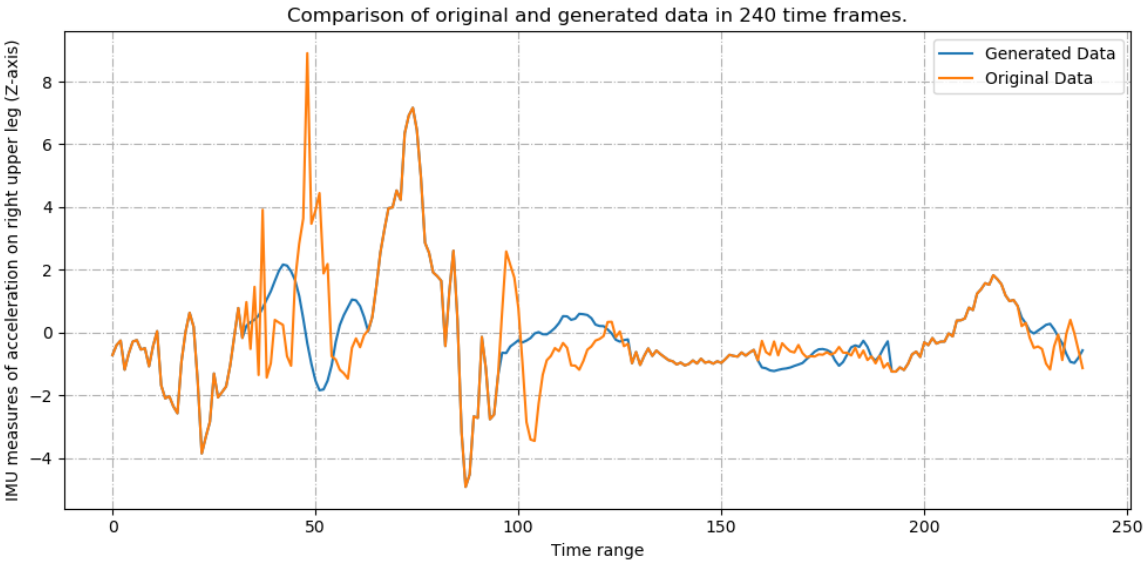


Figure A.8: Comparison of the original and the generated IMU measures of the acceleration part from Imputer in MisGAN on the right upper leg (Z-axis) for 240 time frames (Dataset I).

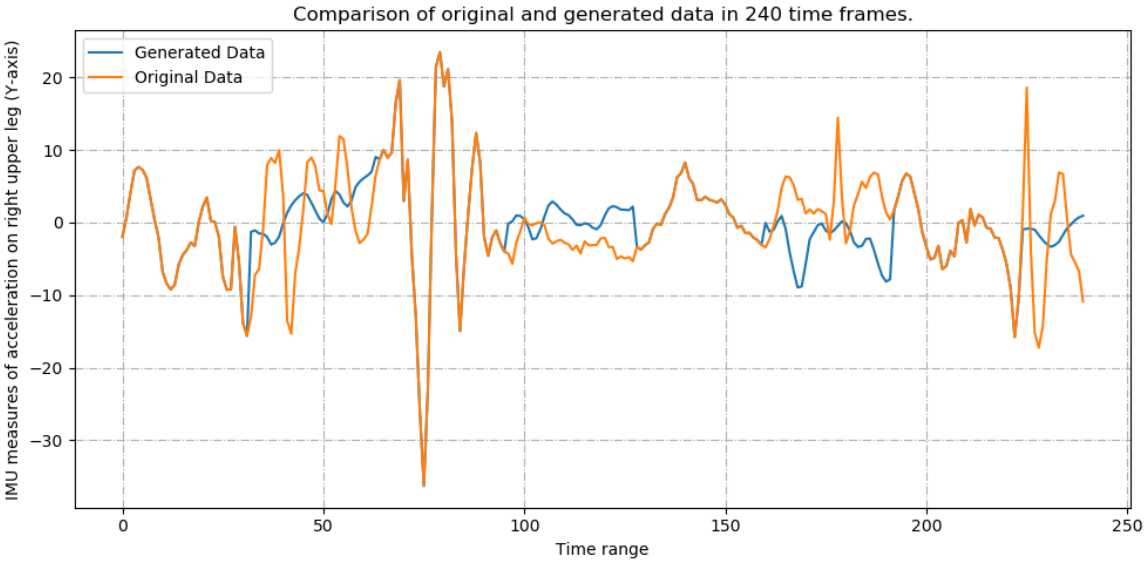


Figure A.9: Comparison of the original and the generated IMU measures of the acceleration part from Imputer in MisGAN on the right upper leg (Y-axis) for 240 time frames (Dataset II).

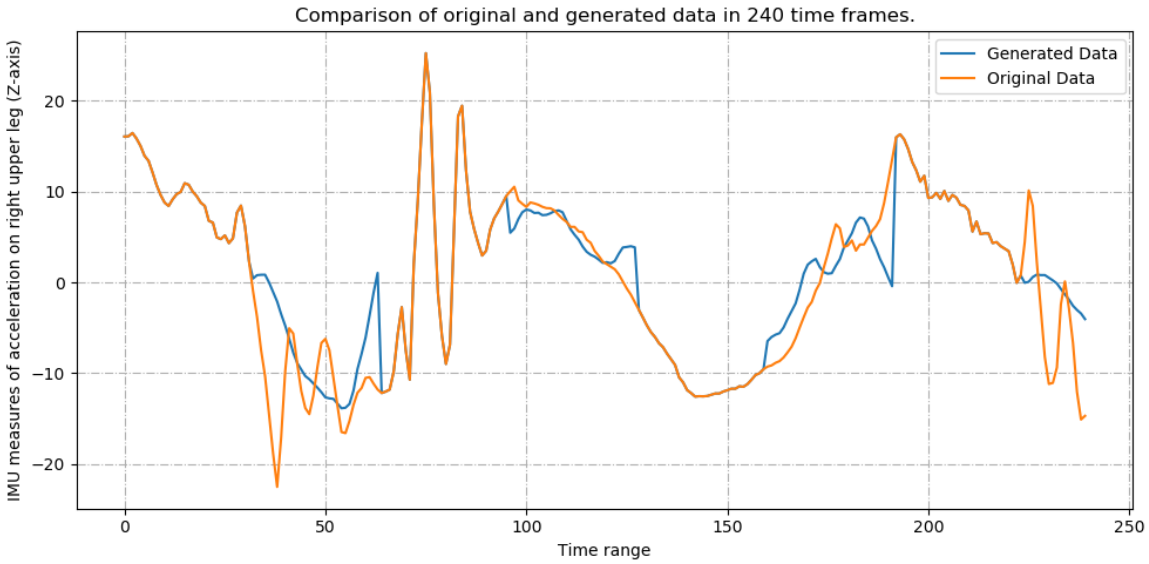


Figure A.10: Comparison of the original and the generated IMU measures of the acceleration part from Imputer in MisGAN on the right upper leg (Y-axis) for 240 time frames (Dataset II).