Summer 2012

Artificial neural network for studying human performance

Mohammad Hindi Bataineh
University of Iowa

ARTIFICIAL NEURAL NETWORK FOR STUDYING HUMAN PERFORMANCE

by

Mohammad Hindi Bataineh

A thesis submitted in partial fulfillment
of the requirements for the Master of
Science degree in Biomedical Engineering
in the Graduate College of
The University of Iowa

July 2012

Thesis Supervisors: Professor Karim Abdel-Malek
Dr. Timothy Marler

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

_____

MASTER'S THESIS

_____

This is to certify that the Master's thesis of

Mohammad Hindi Bataineh

has been approved by the Examining Committee
for the thesis requirement for the Master of Science
degree in Biomedical Engineering at the July 2012 graduation.

Thesis Committee: _____
Karim Abdel-Malek, Thesis Supervisor

_____
Timothy Marler, Thesis Supervisor

_____
Jasbir Arora

_____
Soura Dasgupta

_____
Salam Rahmatalla

_____
Thomas Schnell

To my parents, family, and friends

Innovation distinguishes between a leader and a follower

Steve Jobs

ACKNOWLEDGMENTS

I am grateful to my research advisor, Dr. Timothy Marler, for his patience and contributions in teaching me and helping me to present my ideas clearly. I am especially grateful to my academic advisor and mentor, Professor Karim Abdel-Malek, who directed and encouraged me to work on exciting and appropriate topics. In addition, I would like to thank my other thesis committee members for their time and useful feedback. I would also like to thank Melanie Laverman, Dr. Rajan Bhatt, Anith Mathai, and all Virtual Soldier Research group for their efforts. Finally, above all, I am grateful to the God (firstly and lastly), and to my Mom and Dad for their confidence and support.

ABSTRACT

The vast majority of products and processes in industry and academia require human interaction. Thus, digital human models (DHMs) are becoming critical for improved designs, injury prevention, and a better understanding of human behavior. Although many capabilities in the DHM field continue to mature, there are still many opportunities for improvement, especially with respect to posture- and motion-prediction. Thus, this thesis investigates the use of artificial neural network (ANN) for improving predictive capabilities and for better understanding how and why human behave the way they do.

With respect to motion prediction, one of the most challenging opportunities for improvement concerns computation speed. Especially, when considering dynamic motion prediction, the underlying optimization problems can be large and computationally complex. Even though the current optimization-based tools for predicting human posture are relatively fast and accurate and thus do not require as much improvement, posture prediction in general is a more tractable problem than motion prediction and can provide a test bead that can shed light on potential issues with motion prediction. Thus, I investigate the use of ANN with posture prediction in order to discover potential issues. In addition, directly using ANN with posture prediction provides a preliminary step towards using ANN to predict the most appropriate combination of performance measures (PMs) - what drives human behavior. The PMs, which are the cost functions that are minimized in the posture prediction problem, are typically selected manually depending on the task. This is perhaps the most significant

impediment when using posture prediction. How does the user know which PMs should be used? Neural networks provide tools for solving this problem.

This thesis hypothesizes that the ANN can be trained to predict human motion quickly and accurately, to predict human posture (while considering external forces), and to determine the most appropriate combination of PM(s) for posture prediction. Such capabilities will in turn provide a new tool for studying human behavior. Based on initial experimentation, the general regression neural network (GRNN) was found to be the most effective type of ANN for DHM applications. A semi-automated methodology was developed to ease network construction, training and testing processes, and network parameters. This in turn facilitates use with DHM applications.

With regards to motion prediction, use of ANN was successful. The results showed that the calculation time was reduced from 1 to 40 minutes, to a fraction of a second without reducing accuracy. With regards to posture prediction, ANN was again found to be effective. However, potential issues with certain motion-prediction tasks were discovered and shed light on necessary future development with ANNs. Finally, a decision engine was developed using GRNN for automatically selecting four human PMs, and was shown to be very effective. In order to train this new approach, a novel optimization formulation was used to extract PM weights from pre-existing motion-capture data. Eventually, this work will lead to automatically and realistically driving predictive DHMs in a general virtual environment.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

A digital human model (DHM) is a human representation on computer software used to perform analyses and evaluations related to human performance. This performance includes human posture prediction, motion prediction, ability to complete a task, workplace design, and many other ergonomics studies that concern what is done by human beings.

Artificial neural network (ANN) is a mathematical model for predicting system performance (i.e., system output) inspired by the structure and function of human biological neural networks. The ANN is developed and derived to have a function similar to the human brain by memorizing and learning various tasks and behaving accordingly. It is trained to predict specific behavior and to remember that behavior in the future like the human brain does. Its architecture also is similar to human neuron layers in the brain as far as functionality and inter-neuron connection. ANN has been successfully used in various applications, including those in the DHM world.

ANN is a type of multi-dimensional regression analysis models. It is better in some way than the other regression models, because ANN is more powerful in solving practical and complex problems. Generally, researchers apply and use ANN in system prediction problems when: 1) known and reliable system input/output sets are available (i.e., training data availability), 2) fast system prediction is required, and 3) the system is complicated and difficult to express in mathematical formulas. In general, the ANN is able to predict any system accurately and rapidly no matter how complex the system.

This thesis works on using ANN to predict applications in the DHM field. First, the proper type of ANN for the DHM applications was selected. Next, a new semi-automated strategy was developed to ease the network construction, training and testing processes, and network parameters. Then, the new strategy was used to predict some

task-based human motion and posture predictions using ANN. Finally, a decision engine was developed using ANN for selecting human performance measures (PMs), which are functions that control human performance when intending to perform a task.

This chapter presents an introduction to the thesis work and provides background about the general use of ANN and DHM applications. In addition, it describes a hypothesis and the motivation for targeting the use of ANN in predicting DHM. The chapter starts by introducing background on the general research concepts for this thesis. Then, the research motivation and literature review are presented to show the scope of the current state of the art in the DHM field and the various applications of ANNs. Finally, the research objectives and contributions are presented, followed by a brief overview of the thesis.

1.1. Background

In virtual human science, it is important to study and predict human performance, which includes motion, posture, grasping, etc., in order to duplicate realistic human performance. Human motion prediction is to predict a human's motion during specific tasks like walking, running, jumping, etc. This prediction is achieved by an optimization-based method in which the body's degrees of freedom (DOFs) are the design variables to achieve the required motion, and the PMs are the functions to be minimized. Similarly, human posture prediction is a prediction of a human's posture while performing a defined task like reaching a point, lifting a box, etc. Human performance prediction, however, is not easy because it is influenced by PMs, which are the cost functions that a human tries to minimize to perform such a task. Depending on the task, one or more of those PMs are minimized. For example, minimizing the joint displacement PM alone was useful for touching points in front of the body, but it provided bad postures for points behind the body; it should combine with other PMs like potential energy. Maximum joint torque PM was used to provide realistic postures in box-lifting studies (Marler et al., 2011).

Generally speaking, PMs are the functions to be minimized in single-objective optimization (one PM) or multi-objective optimization (two or more PMs) by finding the body DOFs, which are the design variables of the optimization problem. The way these PMs are combined in different human performances is still unclear because there are many kinds of PMs, and there are various tasks to be studied. In addition, which PM to include for a task and the weight or importance of that included PM in the optimization problem has not yet been studied. Researchers also found that PMs are highly correlated; changing their values or types affects performance results.

In order to develop successful DHM, researchers have worked on various kinds of human actions (motion, posture, etc.). Predicting human motion and posture is important for understanding what drives human performance, and is useful in other practical studies like human simulations. The most advanced and successful works were in human posture prediction, where researchers developed different approaches to apply and predict human postures on computer human models. This work on posture prediction is mature in terms of predicting human postures in real time accurately and for all body joints. On the other hand, researchers are trying to make posture prediction more robust and adaptable to the task to be accomplished.

Unlike posture prediction, human motion prediction is very complex because it requires dynamic prediction for body DOFs. Solid work was done toward developing a high-fidelity human motion prediction that could perform in a task-based manner like the predictive dynamic algorithm that was introduced by Xiang et al. (2010). On the other hand, those available algorithms need a long time, averaging in minutes, to run and provide the motion for such a task; processing time is needed to calculate and optimize different DOF values over the motion time for such a task. The motion prediction calculations are a problem even for simple tasks or rerunning the same task with some minor input changes. Hence, there is a need for real-time motion prediction in order to allow the user to see immediate results for changes in any task input parameters.

Moreover, more reliable sources for human motion prediction, like motion capture data, should be used in order to obtain robust results and practical indications about how and why humans perform.

In this thesis, a new ANN-based approach is presented to study some DHM applications, which could eventually lead to understanding how humans think or behave when performing tasks. These applications include: 1) motion prediction, 2) posture prediction, and 3) performance measures (PMs) determination. The new approach investigates improving the speed of computations in the motion prediction task and making posture prediction more robust. In addition, the thesis tries to address determination of PMs using ANN. As mentioned, humans try to minimize some PMs during tasks. The PMs could be joint-based, like maximum joint torques, displacement, and discomfort, or energy-based. Studies were done using one or some of those PMs in posture prediction to get ideas about their effects on human performance. The studies, however, minimally specify when and why to use such PMs, and for which kind of tasks (Marler et al. (2005a). Therefore, predicting the use of different PMs at various tasks is critical to get a better understanding of human performance and a better representation of what drives human performance.

The work in this thesis focuses on using ANN to initiate a pathway for solving problems related to human performance prediction. The ANN can predict complex systems quickly for best system results and shows excellent trade-off between the input parameters or factors that control the system. This ability is achieved because the network depends on some training or learning data (i.e., known sets of system input and output), like the human brain does. So, the network predicts any new input by doing minor calculations because it was trained to see similar inputs. The learning process needs experience to perform on the network for best prediction, because some network parameters need to be set depending on the predicted system’s inputs and outputs. More detailed information about ANN, its architecture, and its learning process will be

described in Chapter 2. The learning process and network parameters would be automated so that any user could use it, including those who are not expert in ANN. Currently, the common limitation for using different types of ANNs is that some of the parameters are determined by trial and error, which could produce bad predictions. Moreover, there are different types of ANN, such as the feed-forward neural network (FFNN) and the radial basis neural network (RBNN), that are used to solve problems similar to those in DHM; these types have other subtypes under their names. Therefore, the best type of ANN for DHM applications has not been determined; it depends on researcher preference. Different types of FFNN and RBNN were used in many applications with varying success rates.

1.2. Literature Review

The current DHM needs and state of the art in ANN applications need to be reviewed. This section discusses the literature regarding the broad use of ANNs in various applications, especially in DHM fields, with regard to their ability and the kind of problems they can handle. Three DHM applications that target understanding human intelligent behavior are also discussed. Those applications include human motion prediction, posture prediction, and joint-based PMs.

1.2.1 Artificial neural network (ANN) applications

ANN is fast and accurate because after the training process is completed, optimization and time-consuming calculations are no longer needed. So, the network outputs are predicted directly for the provided inputs based on what it has learned to predict for a specific system. There are many ANN types that are used for various applications such as engineering, weather and flood forecasting, business, and medicine because of their power and ability to generalize any practical problem (Coit et al., 1998; Twomey et al., 1998).

Generally, ANN applications fall into the categories of data clustering, classification, or regression. Data clustering creates relationships between fed inputs and separates them into different clusters based on their similarities. In classification, inputs are assigned to their class among different classes. Data regression means creating a curve that passes and fits between training data sets. The data regression type is normally used to predict and solve DHM applications. The main regression types of ANN are FFNN and RBNN, both of which have other subtypes under different names. The ANN detailed architecture and common types will be described in Chapter 2. Variables that could be used as input parameters in DHM applications include, but are not limited to, human anthropometry, the task to be performed, load existence, position (sitting or standing), joint ranges of motion (ROMs), and model DOFs.

Researchers incorporate ANN when they want to save time or cost in system development, or when they are unable to represent the system with a mathematical algorithm. For example, ANN was used to find the Cobb angle, which indicates scoliosis severity, by selecting the optimal set of input torso indices (Jaremko et al., 2002). The Cobb angle (ANN output) was calculated with accepted accuracy. Tani et al. (2008) trained a recurrent neural network (a type of ANN) on a humanoid robot to learn to manipulate objects. The results showed that the network can afford both generalization and context dependency in generating skilled behaviors. In addition, ANN was used in linguistics by Collobert et al. (2008) for language processing predictions. For a given sentence (ANN input), they trained the network to predict part-of-speech tags, chunks, named entity tags, semantic roles, semantically similar words, and the likelihood that the sentence makes sense.

Recently, researchers became more interested in using ANN in DHM because of its efficiency and accuracy in solving problems like human performance prediction. Such studies applied ANN in humanoid motion prediction, obstacle avoidance, human posture prediction, and other human-workplace problems. Zha et al. (2003) proposed an ANN-

based approach for human-machine system design and simulation that predicts an operator's postures and joint angles of motion (ANN outputs) associated with a range of workstation configurations (ANN inputs). Another study was done by Li et al. (2007) on humanoid dynamic obstacle avoidance. The idea was to use cameras to collect the motion path of dynamic obstacles (ANN input) and then build a prediction model using a radial basis neural network (RBNN) using those data. Dynamic obstacles can be utilized in local path planning for a mobile robot in dynamic and uncertainty environments. Once the planner finds there is a dynamic obstacle in the rolling window, the network predicts the obstacle's motion path (ANN output) in the next period based on three time sequence values of the obstacle in a continuous period of time. Work-related posture prediction and human-machine work efficiency using RBNN were presented by Zhao et al. (2010); the RBNN was used in mapping posture prediction and also in referring to the working efficiency with around 27 DOF. That study concluded that RBNN was fast in terms of calculating the virtual human postures and promising in solving ergonomics simulation and assessment of human-machine system problems. Those works are examples that show how efficient ANN is in handling practical problems, especially in DHM studies. Grasping tasks were also tested using ANN by, for example, training the network on specific grasps corresponding to finger positions. Inverse kinematics mapping between the fingertip 3D position and the corresponding joint angles were proposed and evaluated using ANN (Rezzoug et al., 2008), where the network had 3 inputs and 21 outputs. The study used an instrumented glove for mapping finger movements to a multi-chain model of the hand. From the fingertip desired position, the network allowed predicting the corresponding finger joint angles that achieve finger positions.

Applying ANN in broad fields shows its ability in solving different kinds of problems. Scholars have used ANNs in DHM problems and obtained acceptable results, but no one type of ANN has been determined to be best for DHM problems. One study claimed that RBNN was fast in terms of calculating the virtual human postures and might

be good for similar applications, but in reality, all ANNs work quickly once the training is completed. The network types, however, differ in speed of training and time for optimizing or learning their prediction capabilities during the training phase. For example, FFNN experiences memory problems if the number of inputs and/or outputs is relatively large, and this limits using it for some applications, especially in DHM. In human motion prediction, the number of predicted outputs is large because that includes predicting all body DOFs and their values at all time frames over the motion task. That means hundreds of outputs, which might explain the limited use of FFNN and ANNs in general for studying human performance applications. Moreover, the learning process, which is the core of using ANN, should be performed in a more generic and smarter way so that any system is predicted in the proper way.

Within the context of human modeling problems, ANNs are applied only to very specific scenarios and have not yet been developed for robust use with more general DHM problems. To date, ANNs have been used in DHM for solving confined systems with a small number of parameters or conditions. The following discussion will present the current state in three DHM applications with the focus on using ANN as an approach in these applications. These applications include motion prediction, posture prediction, and PMs.

1.2.2 Human motion prediction and artificial neural network (ANN)

As part of studying human biomechanics and DHM, it is important to understand human motion as a dynamic function and a way to touch the factors or drivers that direct human thinking in task performance. Real understanding and duplication of human motion strategies also inspires many industrial fields that use intelligent moving parts as well as humanoid dynamics. Most motion prediction studies depend on motion capture systems as an approach to track, record, and reproduce human motions. Chaffin et al.

(2001) used a motion capture system to record human motion while reaching and moving light to moderate load objects while either seated or standing. Then, they used a 17-link kinematics model to resolve the dynamics of the motions where their initial motion prediction algorithms captured well over 95%. Other various optimization and mathematical approaches are also used to reconstruct human motion that is real and computationally fast. Xiang et al. (2010) developed the predictive dynamic (PD), which is a simulation for human motion based on formulating an optimization problem with appropriate PMs and constraints that depend on the predicted task. Despite the PD novelty in reproducing realistic motions, computational speed is still the main limitation to making the PD work in real time.

Another efficient mathematical approach for predicting human motion is ANN, which can handle complex problems like human motion. Researchers applied ANN in gait and motion analysis; the FFNN type was used to manipulate an electro-myogram (EMG) signal and joint motion in predicting a joint's stiffness control strategy during a specific contact task (Kang et al., 2007). For that study, they tested three different ANNs. The network inputs were 16, 18, and 20, respectively, for all proposed networks with four outputs. The results showed that the third network gave the most accurate results without any problems in terms of time or training processes for using that number of inputs. The third network provided the best results because it had more parameters that described the problem more specifically, which improved the prediction ability. Najmaei et al. (2010) showed that by using FFNN, the future motion trajectory of the human can be integrated in a reactive control safety strategy to foresee and react to an upcoming dangerous situation prior to its occurrence. As a result, it was shown that applying a prediction-based reactive control strategy using ANN could compensate for the delay due to danger evaluations and the modification of the path and could significantly improve the performance.

Moreover, the efficiency of using ANN for motion prediction is shown clearly in robotics applications. Hahn et al. (2005) performed a study to demonstrate the effectiveness of ANN in mapping gait measurements into motion. The FFNN type was used in that study and improved the performance by minimizing the processing time and increasing the accuracy of the mapped motion. Nishide et al. (2009) developed a model that searches and generates robot motions using two types of ANN. They trained the recurrent type of neural network with parametric bias to self-organize robot and object dynamics. Then, another type of ANN, the hierarchical neural network, was trained to link the object image with the searched motion. The results showed that the robot acquired the most reliable motion and shifted it according to the posture of the object. Bu et al. (2009) also proposed a task model motion prediction using a Bayesian network type. That model was able to predict occurrence probabilities of the motions concerned in the task by using information from the previous motion.

Some work showed that incorporating ANN in motion prediction improved the accuracy and computational speed, which was responsible for the delay in some outputs when using other methods. One of the studies compared three ANNs' prediction abilities for human motion performance and found that the one with the most input parameters had the best output ability. This indicates that expressing a system using more inputs will improve the general performance in terms of handling more conditions as well as predicted outputs. Most of the work discussed here, however, used FFNN, which has some memory and accuracy issues with a large number of inputs and outputs. This suggests that other types of ANN should be tested and used for motion prediction. In addition, no work has been done on prediction of motion for a full human model, likely because of limitations in the types of ANN used. Moreover, no one has coupled ANN with PD to improve the speed of calculations.

1.2.3 Human posture prediction and artificial neural network (ANN)

Besides human motion prediction, posture prediction is an important part of designing a virtual human. A lot of effort is required to find the best way to predict and express any human model's posture. We need to predict posture for many ergonomic studies as well as human-machine workplace design. Predicting real posture is complicated because humans behave differently and choose their postures based on many factors that are still unclear. There are two main approaches to solving posture prediction problems. The first approach involves prerecorded data using motion capture systems combined with anthropometric data and functional regression models (Beck et al., 1992; Chaffin et al., 2001). Those recorded data are used to build an algorithm that simulates different human motions (Park et al., 2002). Methods such as the pseudo-inverse method are then used to solve the optimization problem and find the best expected posture from the robot or virtual human model (Liegeois et al., 1977). The prerecorded data method, however, is not able to provide a wide range of creative or hypothetical postures.

The second approach involves real-time inverse kinematic optimization-based posture prediction that has recently been introduced as a dependable way to predict human posture based on some objective functions. It depends on defining the final position (target point) that needs to be reached. Basically, the optimization problem has three main parts: (1) variables to be found, (2) objective functions, which are functions of those variables, and (3) constraints, which are the limits that the design can't exceed. For human posture prediction, joint angles are the design variables. Many factors are incorporated in human posture prediction as objective functions, called PMs, that play an important role in brain decisions. Some of those factors are related to comfort level, like moving near joint limits, while others are related to the tendency to minimize joint torques, potential energy, joint displacements, and visual displacement. The constraints include the distance between the last link end and the target point, as well as the upper

and lower joint ROM limits. Riffard et al. (1996) found the optimum torso and arm displacements with seven DOF using an unconstrained optimization approach. Abdel-Malek et al. (2001) developed an efficient numerical formulation for the prediction of real postures, which was based on kinematics for modeling with optimization of a cost function to predict a realistic posture.

Whether human posture prediction is obtained from inverse kinematics or prerecorded data, some generalization methods like ANN have been used to handle some posture prediction problems and recreate postures in a fast and accurate manner. Researchers found that ANN is applicable to predicting posture without significant difference from the traditionally used methods (Jung et al., 1994). In that study, FFNN was trained to predict a 9-DOF human arm (DOF values were the outputs), and it provided acceptable results. There was no significant difference between the coordinates of the joints generated by the network and those measured from human posture reaching for the given targets (ANN inputs). Perez et al. (2008) used FFNN to predict two lifting postures based on finding joint angles that represent these postures. The network had 7 inputs and 10 outputs; it predicted whole-body posture with an error of 5-20 degrees per joint angle for most body angles. Zhang et al. (2010) used FFNN to predict posture where the network inputs were landmarks that characterize human posture while the predicted outputs represented the transformed posture, which is a set of other landmarks. The range of errors in ANN prediction was acceptable in posture prediction, because it is hard to define the best posture for accomplishing a specific task since people just behave differently.

The use of ANN in human posture prediction was developed early and applied in various studies. All studies referred to the potential use of ANNs to predict real and fast human postures. Application of ANNs in posture prediction is also powered by the available and reliable sources for training the network on intended postures. Those current posture prediction sources could predict postures in real time. On the other hand,

the use of ANN in human posture prediction in the context of a full human model (i.e., a human model with a realistic number of DOFs) is still unavailable. Like human motion prediction, studying human posture prediction using ANNs needs to be done by different types of ANNs, which might improve the prediction ability and handle the large number of DOFs for such a human model.

1.2.4 Joint-based performance measures (PMs)

As stated earlier in the background, motion and posture prediction problems are affected by many factors and cost functions called PMs. These PMs could be joint-based, like maximum joint torques, displacement, and discomfort, or whole-body energy-based, like potential energy. Completing a task based on optimizing those PMs is a multi-objective optimization (MOO) problem, where the PMs are optimized together in one function. In general, PMs control human decisions about motion, posture, sitting, standing, and any other task to be accomplished. In this thesis, four joint-based PMs are studied, including: 1) discomfort, 2) joint displacement, 3) maximum joint torque, and 4) total joint torques. We focus in this thesis on the use of these four PMs in the context of a human posture prediction problem. The PMs are differently combined in the optimization problem depending on the task and the conditions of that task. The MOO problem is formulated as follows (Marler et al., 2004):

$$\begin{aligned} &\text{Find:} && q \in R^{DOF} \\ &\text{To minimize:} && f(q) = [f1(q1) \quad f2(q2) \ldots . \quad fk(qk)]^T \\ &\text{Subject to:} && q_i^L \leq q_i \leq q_i^U; i = 1, \ldots . , n \\ & && \|x(q) - p\| \leq \varepsilon \end{aligned} \tag{1.1}$$

where: q: joint angle

$[f1(q1) \ f2(q2) \ldots . fk(qk)]^T$: cost functions (i.e., PMs) to be minimized.

q_i^L, q_i^U: the lower and upper joint angle limits, respectively.

$x(q)$: position of vector in Cartesian space that describes the location of the end-effector as a function of the joint angles, with respect to the global coordinate system.

ε: a small number that approximates zero

p: the position vector of the target point

Joint displacement is one of the major PMs to obtain real human posture prediction. This PM is incorporated to include different costs for moving various body joints, depending on which joints the human likes to move in his posture. For example, when we need to reach a point in front of our bodies, we first try to reach it by moving the hand alone. Then, if the point can't be reached, we move the shoulder with the hand and then the torso if it is still unreachable. Marler et al. (2005a) showed that joint displacement, even if it is used alone in the optimization problem, provides subjectively acceptable results for touching most points in front of the body. The discomfort PM used in this thesis was developed by Marler et al. (2005b). The two types of joint-torque-based PMs that are used in this thesis are: 1) the sum of joint torques (total joint torques) and 2) maximum joint torque (Marler et al., 2011; Liu et al., 2009). Generally, joint torque represents the force of a group of muscles acting on or around a joint, and the torque limits represent strength. Actually, both torque-based PMs are different in their contributions in posture prediction and provide different results when they are used separately.

The literature referred to and used the PMs either individually or in groups of two or more. Scholars showed that MOO is the most practical and real option for studying human posture prediction (Yu et al., 2001; Mi et al., 2004). Basically, in humans there are many PMs that measure the human's performance to complete posture in a task-based manner. Some of them are still under study, such as joint fatigue, which is hard to determine. Others, however, have received more interest in the literature, like discomfort, potential energy, and joint displacement. Marler et al. (2005a) also studied joint displacement and potential energy in posture prediction, and concluded that potential

energy cannot be used alone in posture prediction. It should be used in small weight (i.e., less weight value in the MOO problem) with other objective functions, and it is more helpful for predicting real postures at target points located behind the body. Another study was introduced for fatigue effect on MOO in posture prediction, and the changes in predicted posture results were analyzed mathematically with and without fatigue existence (Ma et al., 2009). Recently, other MOO method studies were done for using three or more PMs (Marler et al., 2010; Yang et al., 2010). The remaining question is how to choose those different PMs for any posture prediction task.

In summary, studying human PMs should go along with posture prediction because proper use of different PMs in various posture tasks produces robust and real postures, therefore increasing understanding of how the human behaves when performing. The scholars above have studied posture prediction and other ergonomic fields using various PMs. This inconsistency of choosing PMs in the studies indicates the need for thorough analysis among those PMs to generalize their use for any task. The literature succeeds in extracting some general trends for using particular PMs. For example, minimizing joint displacement was good in posture prediction for points in front, while potential energy should be combined in small fraction with some other PMs. These conclusions could be enhanced by using ANN, but its use has been very limited and exclusive to some general human predicted posture.

1.3. Motivation and Hypothesis

Important work has been done on DHM fields using various approaches. Many ANN types had advantages in improving system performance for broad applications, including some for DHM. This thesis tries to embed ANN in solving some DHM problems and overcoming some of the current limitations. This work is motivated by the following:

1. Within the context of human modeling problems, ANNs are applied only to very specific scenarios and have not yet been developed for robust use with more general DHM problems. To date, ANNs have been used in DHM for solving confined systems with a small number of parameters or conditions.
2. The best type of ANN to be used to predict DHM applications has not been determined. Different types of FFNN and RBNN were used in many applications with varying success rates. The RBNN type of ANN, however, is fast in terms of calculations during the training process and can handle a larger number of inputs and outputs than FFNN.
3. There has been no work on using ANN for motion and posture predictions for a full human model, likely because of limitations in the types of ANN used. The previously used types of ANN in DHM field experienced memory and training problems when these networks predicted DHM with relatively large number of DOFs.
4. ANN has never coupled with PD to improve the speed of calculations. The computational speed is still large, averaging in minutes, which is a problem even for simple tasks or rerunning the same task with some minor input changes.
5. There has been no work toward proper use of combination of various PMs in posture prediction problems in a task-based manner.

In general, there is potential for improving predictive DHMs using ANN, but DHMs will require use of a specific type of ANN. Successfully implementing and predicting DHM using ANN means that it handles the human performance decisions that are done by the human brain. Eventually, work on the level of ANN's parameters and properties should reveal the functionality of the natural neural system in the brain, which regulates the human performance. Specifically, the following hypotheses will be tested:

1. ANN can be integrated with PD and can increase the computational speed of PD without detriment to accuracy.

2. Despite the potential complexity of ANN, it is possible to develop software and an associated methodology for intuitive and easy use.
3. ANN provides a method for automatically determining the most appropriate PM(s) for a general set of tasks.

1.4. Objectives

The presented hypotheses are examined in this thesis by trying to choose the best type of ANNs for enhancing the DHM predictive applications. Modifying the network construction and training is needed for optimal network performance. Then, the selected network should be evaluated by applying it to solve the current limitations in the DHM applications. Technically, solving these limitations should lead to understanding the general trend of human behavior when performing any task. This work pursues the following objectives:

1. Select the best candidate network type of ANN to be used in DHM applications based on the advantages of the network and DHM needs (i.e., the ANN that works well for real and complete DHM problems).
2. Automate the training process and set the network properties for the selected network to ease the use of that network in DHM by any user, and for any application.
3. Generalize the use of ANN to predict relatively large DHM problems, where the selected network should be able to predict a relatively large number of outputs, in hundreds, from different types like joint angles, torques, ground reaction force, etc. By generalizing the prediction capabilities, the network provides this large number of outputs in a task-based manner under various conditions.
4. Try to use the selected network for predicting a large number of DOFs in motion and posture problems, and from different types like those mentioned in objective number 3.

5. Investigate using the selected network to speed up the computations of the predicted motion task, and couple the network with PD.
6. Develop a decision engine to select the proper PMs in a posture prediction problem in a task-based manner.

1.5. Thesis Overview

After an introduction to the current state of the art in the DHM field and to ANN as a potential approach to solve the limitations in this field, the second chapter describes ANN architecture in detail and discusses the most common types. The third chapter presents the new methodologies that this research provides, including proposed strategies for better use of ANN in DHM fields. The fourth chapter presents two applications for the use of the best selected network and the new proposed strategies in the context of motion prediction. The fifth chapter presents the use of the same network and strategies for predicting two posture tasks. Then, the sixth chapter presents the development of a decision engine to determine the PMs using the same selected network. A new training method is also presented in that chapter. The last chapter discusses the general achievements of this research, including summary, discussion, and future and long-term work that could build on the accomplished work.

CHAPTER II

ARTIFICIAL NEURAL NETWORK: BACKGROUND

The artificial neural network (ANN) is the proposed approach to be used in this thesis to solve the current limitations in digital human modeling (DHM) applications and to examine the validation of the hypothesis. In addition, part of this thesis concerns developing a semiautomatic technique for training ANN, as well as automatically selecting its properties for the best system prediction. Hence, this chapter first talks about the basic concept of ANN and its architecture, and then describes the network's training process. Then, some of the most commonly used ANNs are described as far as their advantages, disadvantages, and applications. The final section focuses on the mathematical details of the selected ANN type that will be used in this thesis.

2.1. Basic Concepts

The human brain is a decision system in the human being that has millions of neuron folds connected in a complicated way. The brain is more powerful and faster than any computer processor in handling complicated problems related to human performance problems. The human brain has multiple layers of neurons that interact with each other in parallel. The parallel interaction means that each neuron receives input lines from all neurons in the previous layer and sends different output lines to all neurons in the next layer. The neuron also sends values to the previous layer and receives values from the next layer. By learning and memorizing doing such a task, these received and sent values to and from the neurons are set for that task to provide the proper decision.

The brain has powerful decision abilities to solve various complex problems like motion, posture, mathematical calculations, etc. This ability is gained by memorizing and learning previous cases that are similar to these problems. The literature mimicked the brain's neuron architecture to duplicate the underlying functionality of the brain, which

depends on the training and memorizing to solve various problems and perform different tasks under varying conditions. The new mimicked approach is called ANN, which is successfully used to predict many practical problems.

ANN is an intelligent mathematical algorithm that consists of three main parts: (1) input layer, (2) middle or hidden layer(s), and (3) output layer. First, the input layer consists of the system's inputs. The "input layer" is simply a vector of the inputs. Second, the hidden layer represents the core of the ANN and consists of many units called neurons. Inside the neurons, the main mathematical calculations occur to process the inputs and provide the proper outputs. Like a biological neuron in the real brain, the neuron in the hidden layer receives and sends a line of values from the previous layer and to the next layer, respectively. The received and sent values to and from the neuron differ depending on the weight value of the channel (i.e., line) that carries the value to and from the neuron. The weight of the channel means the value that is multiplied by the carried value (i.e., multiplying the weights value by the coming value from the previous neuron) before passing the result to the next neuron. The weight value changes by changing the intended task to perform; its value is decided by learning and memorizing doing that task. Figure 2.1 shows a simple feed-forward neural network (FFNN) type of ANN.

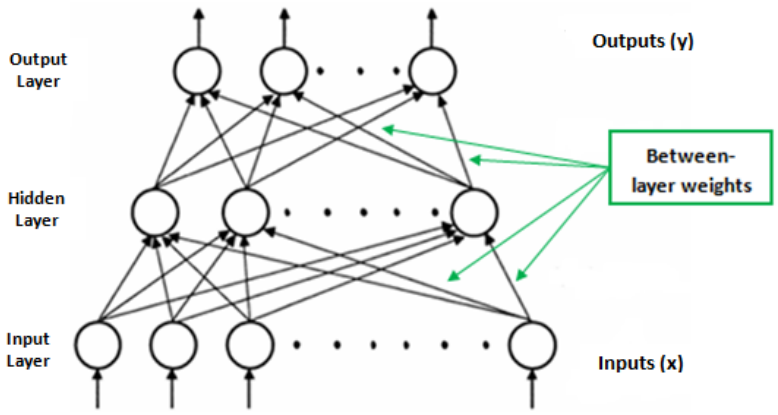

Figure 2.1: Simple feed-forward neural network.

In Figure 2.1, $\boldsymbol{x}$ represents the vector of the network's inputs, and $\boldsymbol{y}$ is a vector of the network's output. ANN is a kind of unconstrained optimization problem, where the neuron weight's values are design variables to be found. The minimization function is the mean square error (MSE), which is the difference between the network predicted outputs and the exact training outputs. The following shows the optimization formula for the general ANN:

$$\text{Find:} \quad W_i \in R^N, \tag{2.1}$$

$$\text{To min:} \quad MSE = \sum_i^N (T_i - Y_i)^2$$

In the above formula, *N* represents the number of hidden neurons, and W_i is the ith weight value in the "between layer weights." Many models and algorithms are available in ANN, ranging from basic models that could consist of single input, hidden, and output layers to more complex multi-layer ones. The complexity of ANN depends on the problem to be solved. The complexity of such a problem depends on the number of inputs that the network needs to handle and create relationships between and the number of outputs it needs to predict. The network works as a multi-dimensional curve fit (i.e., regression curve) for the system inputs. As an example for the regression curve, Figure 2.2 presents an n-dimensional curve fitting for the training data that is shown as small dots. In the figure, the hidden layer determines the function $f(\boldsymbol{x})$ that expresses the training data, while the neurons determine the dimension of the function (n) and the variable coefficients $[a_0, a_1, a_2, \dots, a_n]$. The system inputs represent$[x = x_2, x_2, x_3, \dots, x_R]$, for (R) the number of inputs. As stated previously, more complex problems need more neurons in the hidden layer and, rarely, more hidden layers, because the literature indicates that a single hidden layer ANN is able to predict any practical nonlinear system (Bishop et al., 1995).

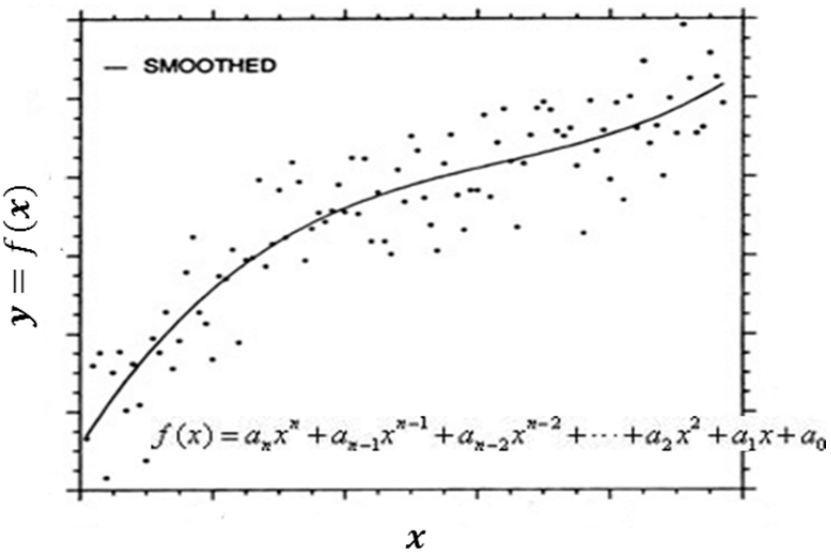

Figure 2.2: n-dimensional curve fitting function.

Basically, building an efficient ANN for system prediction requires careful study of some issues related to the network, including: (1) a universal function approximation capability (i.e., ability to generalize the problem successfully), (2) resistance to noise or missing data (i.e., filtering the noise and extracting the features properly), (3) accommodation of multiple nonlinear variables for unknown interactions, and 4) choosing the proper type of ANN for the best problem prediction (Twomey et al., 1998). Researchers found that it is better to use one hidden layer and work on changing the number of neurons and/or training data sets until the best performance is achieved. The reasons for not using more than one hidden layer are as follows:

1. Adding more hidden layers makes the network performance unstable and subject to more noise, because there are more neurons and connections between the layers.
2. The curve fitting becomes more complex and very specialized to predict the training cases. That specialty reduces the general network prediction ability for inputs other than the training cases.

3. There is more potential to reach a local optimization solution when the network has two or more hidden layers, because there are more neurons to be minimized at different layers. Consequently, the locally optimized network produces wrong outputs.

Equations 2.2 and 2.3, suggest a general way to find the right number of hidden neurons and training cases (Lawrence et al., 1998). However, this is only a good starting point for constructing the network for such a problem. The user would change these numbers during the training process in most applications till the best network performance is achieved.

$$h = (i + o)/2 \tag{2.2}$$

$$3(i - o) \leq N \leq 15(i + o) \tag{2.3}$$

where: *h:* the number of hidden neurons

N: the number of training data sets

i: the number of inputs

o: the number of outputs

2.2. Training Process

During ANN creation and development, the training step is the most important part for proper ANN performance. There are two types of training processes: supervised and unsupervised. The supervised training includes a set of training data where both input and output are known. Generally, it is used in classification and regression problems. In classification, inputs are assigned to their class among different classes. Regression means creating a curve that passes and fits between training data sets. Unsupervised learning is used when inputs are known but the outputs are not, which is the case in clustering problems. The network is trained to create the proper output by combining the inputs in the proper way.

Since DHM applications are regression and prediction problems, a supervised learning process must be used in this study. Hence, the inputs/outputs should be well defined for the targeted task. In addition, a reliable source should be provided to predict exact outputs for the fed inputs. In this thesis, all the above requirements are available by having solid and reliable sources for our models and applications. All these sources and collected training cases (input/output sets) will be described in detail when we talk about the applications of using ANN in the following chapters.

The training process starts by normalization, which is done for the inputs of the training cases. The idea of normalization is to decrease the variance of the inputs vector and compress all inputs into a small range to be handled by the network. The normalization could be done in many ways, where the standard method is to pass the inputs through the sigmoidal function to compress them. In this thesis, we use the standard sigmoidal function, which will be discussed in Chapter 3 (Section 3.2.1).

After the normalization is performed, the next step is to perform the optimization for the network, as in Equation 2.1. The neuron weight's values are optimized and changed during the training process to allow the network to provide the highest achievable accuracy in predicting the provided training cases. MSE is calculated each time the values of the variable are changed until it reaches the accepted minimal value defined by the user. Then, the training process is finished when the required MSE is reached, and the network becomes ready to use for predicting new inputs.

Some common problems occur along with the training process, including: 1) the optimization might stop without reaching the required MSE value and 2) the training process could finish successfully, but the network provides poor outputs when tested with new inputs. These situations mean that the network is unable to handle the relationships between the inputs (i.e., the network curve has lower dimension than what the problem inputs has). Thus, two things should be done to overcome these situations. First, increase the number of neurons in the network and repeat the training process. Adding more

neurons increases the dimension of the curve that fits the training data. If that does not work, it means the current number of training cases is not sufficient for the network to handle (i.e., extract) the relationships between the different systems' features (inputs). Thus, more training cases should be collected before repeating the training process.

2.3. Common Types of Artificial Neural Networks (ANNs)

Many types of ANN have been developed to be used for many applications. Even for the same type, there are ANNs that differ in transfer functions and training approaches. Thus, selecting the most appropriate ANN type for a specific problem is not trivial. In this section, we will talk about the two main types of ANN that are used specifically to solve regression problems, which are the type of DHM applications in this thesis. These ANN types will be presented in terms of their general architectures, advantages, disadvantages, and applications.

2.3.1 Feed-forward neural network (FFNN)

Feed-forward neural network (FFNN), which is shown in Figure 2.3, is one of the most common and first developed types of ANN. Inputs are included in the input layer, which is shown in the figure as a set of circles. The inputs enter the hidden layer by the neuron weights that are shown in the figure. The hidden neurons are represented as circles each inside with a sigmoidal transfer function. The output layer receives the outputs of the hidden layer neurons by another set of neuron weights. Inside each neuron in the output layer, there is linear transfer function, shown in the same figure, to provide the final results (outputs). Generally, the sigmoidal and linear transfer functions are used on the hidden and output layers, respectively, when the problem is a regression type. More information about FFNN architecture and functionality are provided by Bishop et al. (1995).

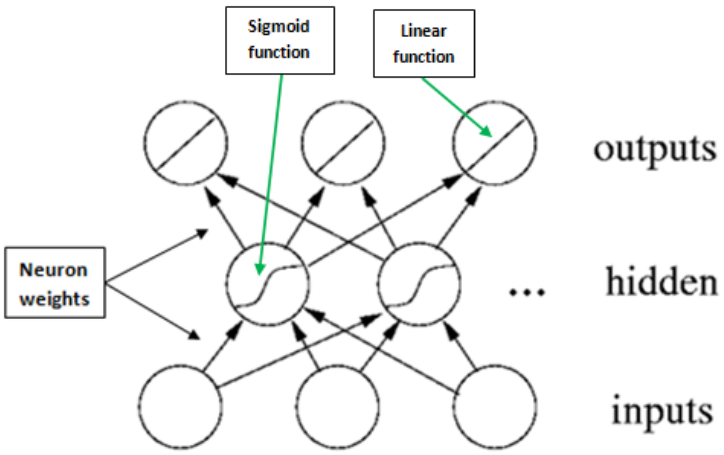

Figure 2.3: Feed forward neural network (FFNN).

FFNN is widely used because of its use in applications in both classifications and regression problems. The advantages of using FFNN are as follows:

1. Generalizing system prediction at any input or extrapolating off-grid training space. After the network is trained, it will be able to predict any new input, even those out of the training limits.
2. Working well for many applications, especially curve fitting of the time series data (i.e., data that come in different times and values).

FFNN, however, has some limitations that constrain using it for some applications. These limitations include the following:

1. It could be highly inaccurate because of local minima solution that comes from optimization. Usually, FFNN has more neurons in its hidden layer than other types of ANN. So, a local optimization solution is more likely to occur in FFNN.
2. It experiences training time and memory issues during the training process because it has more neurons to be optimized.

Therefore, these limitations exclude FFNN as an option in some applications when the number of the training cases and/or inputs and outputs are large. It is also

excluded when high accuracy is required for system performance. Consequently, these limitations drop FFNN as an option for the DHM applications that are presented in this thesis, because they need a network with high accuracy and a large number of outputs.

2.3.2 Radial-basis neural network (RBNN)

Figure 2.4 shows the radial basis neural network (RBNN), which is another type of ANN that is widely used in various applications. Besides the input and output vectors, the network consists of one hidden layer and one outputs layer. Because RBNN provides the foundation for this work, we provide additional details regarding its structure.

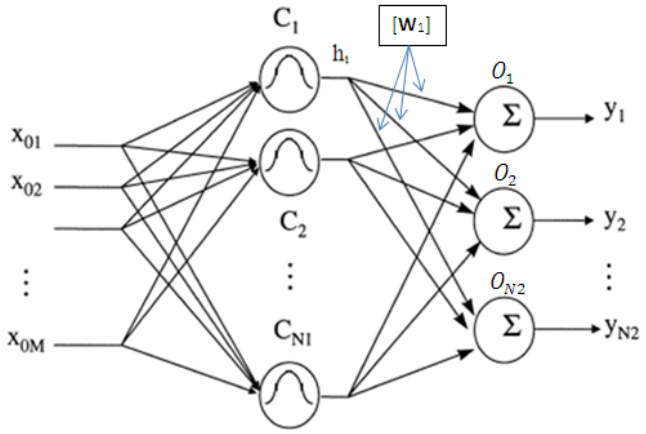

Figure 2.4: RBNN with M-dimensional input and N-dimensional outputs.

In the figure, $\boldsymbol{x} = [x_{01}, x_{02}, \dots, x_{0M}]$ represents inputs of the network, $[C_1, C_{12}, \dots, C_{N1}]$ are the neurons of the hidden layer, $[\boldsymbol{W}_1]$ is the vector of weights at the first neuron in the hidden layer (called line weights), and $[y_1, y_2, \dots, y_{N2}]$ represent the network's outputs.

In the figure, $\boldsymbol{x} = [x_{01}, x_{02}, \dots, x_{0M}]$ provides the input for each neuron in the hidden layer, labeled C_1 in the figure. In this case, the neurons are an essential radial basis function, hence the name radial basis neural network. All of the neurons

collectively constitute the hidden layer. The hidden layer has N_1 neurons $[C_1, C_{12}, \dots, C_{N1}]$. Inside each hidden neuron $C_i(\mathbf{x})$, there is a radial transfer function that produces output h_i. The output h_i is multiplied with weight vector $\boldsymbol{W}_i$ to produce hidden output vector $\boldsymbol{A}_i$. The dimension of the weight matrix, as shown in Equation 2.4, and hidden output matrix $\boldsymbol{A}$ is $N_2 x N_1$. Each row of $\boldsymbol{W}$ and $\boldsymbol{A}$ is referred to as a weight and hidden output vector associated with a corresponding neuron. The output layer has a number of neurons, labeled O_1 in the figure, equal to number of outputs $[y_1, y_2, \dots, y_{N2}]$. Inside each output neuron $O_i(\mathbf{A}_i)$, the output is calculated by taking the sum of the received lines $\boldsymbol{A}_i$, which represents a column of matrix $\boldsymbol{A}$. A full description of this network and its functionality are provided by Buhmann et al. (2003).

$$\boldsymbol{W} = \begin{bmatrix} w_{11}\ w_{12} \dots\ w_{1N2} \\ w_{21}\ w_{22} \dots\ w_{2N2} \\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots \\ w_{N11}\ w_{N12} \dots\ w_{N1N2} \end{bmatrix} \tag{2.4}$$

$$\boldsymbol{h} = [h_1\ h_2 \dots\dots h_{N1}] \tag{2.5}$$

$$\boldsymbol{A} = \boldsymbol{h}^{\boldsymbol{T}} \cdot \boldsymbol{W} \tag{2.6}$$

$$y_i = \sum_{k=1}^{N1} A_{ki} \tag{2.7}$$

RBNN is trained by solving the optimization problem in Equation 2.8.

Find: $\boldsymbol{W_{N1XN2}}$ (2.8)

To min: $MSE = \sum_i^N (T_i - y_i)^2$

In the above formula, T_i represents the ith training output, y_i is the predicted output from the network. Note that y is a function of **W**, as shown in Equation 2.7. The training starts with the first iteration with one hidden neuron (N1=1). Then, N1 is incremented by 1 each time before the next iteration. The optimization stops once the MSE equals a small value (almost zero).

Like FFNN, RBNN is used for all applications in both classification and regression problems. In this thesis, the RBNN is specifically used because of the following superior advantages:

1. It provides highly accurate results within the limits of the training space (i.e., inside the domain of the training values).
2. There are no local minima problems. The network does not optimize to local minimum solutions because the number of hidden neurons is optimized automatically in the training process. Thus, the optimal solution is obtained in terms of the number of neurons and the network weight matrix $\boldsymbol{W}$.
3. There are no computational time and computer memory problems, especially when there are a large number of input/output training sets, because the network does not have a large number of neurons and weights. The weight values to be optimized exist only on the output side of the hidden layer, while FFNN has weights in both sides.
4. It was found by experience that RBNN is the best type of ANN for high-dimensional regression models.

Although RBNN has powerful prediction capabilities, it has some expected limitations, as follows:

1. The network parameter (Gaussian width) is determined heuristically, which could produce poor results.
2. It cannot predict points that are out of training grid space. The network cannot provide accurate outputs when the input is outside the range of training data (i.e., no extrapolation).

There are three different types of RBNN:

1. Exact RBNN.
2. Probabilistic RBNN.
3. Generalized regression neural network (GRNN).

There is no general rule for how to choose the best ANN for a specific problem. Each type is available to be used successfully, depending on the system type, training data availability, system complexity, and desired output accuracy. In this thesis, GRNN is found as the best appropriate type of RBNN to be used for all developments and applications. The GRNN is chosen because of following reasons:

1. It is the most accurate type of RBNN for regression problems, which is the case of most DHM applications, including those presented in this thesis.
2. It does not provide any abnormal results for any point within the training space, because it does not depend on the optimization to adjust the network performance. Thus, no local optimal solution is reached to provide some abnormal (bad) results.
3. It has fewer network parameters that need to be determined heuristically than any other type of ANN. Therefore, it is more stable and easier to construct to obtain the best results.
4. The GRNN training process has no optimization iteration, which allows handling larger problems (larger number of inputs and outputs) without any memory- or training-related problems.

2.3.2.1 General regression neural network (GRNN)

GRNN is considered a special type of RBNN for two reasons. First, it has no iterative optimization in its training process. Second, its hidden layer has a predefined number of neurons equal to the number of training cases. Figure 2.5 shows the general GRNN architecture. In the figure, $\boldsymbol{x} = [x_1, x_2, \dots, x_R]$ provides the input for each neuron in the hidden layer, labeled "**Hidden Layer**" in the figure. The hidden layer has Q neurons $[1, 2, \dots, Q]$. Inside each hidden neuron, there is a radial transfer function that produces output depending on the provided input (Wasserman et al., 1993). The hidden neuron's output enters all neurons in the output layer, labeled "**Output Layer**" in the figure. Each neuron in the output layer essentially combines the received lines (the

outputs of all hidden neurons) in a weighted sum to provide the final network outputs. The output layer has N number of neurons, which is the number of outputs $[y_1, y_2, \dots, y_N]$.

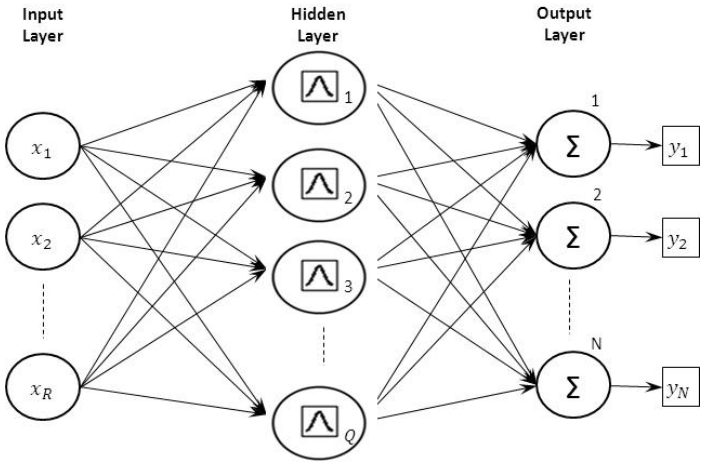

Figure 2.5: General regression neural network architecture.

In Figure 2.5, $[x_1, x_2, \dots, x_R]$ represents the network input parameters, R is the number of inputs, Q is the number of training cases, and N is the number of outputs. $\boldsymbol{x}$ provides input for each neuron in the hidden layer, where some mathematical steps are calculated to provide the outputs to the next layer (the output layer). Figure 2.6 shows a flow chart for the mathematical steps that are calculated inside each neuron in the hidden layer. Once the neuron receives the input **x**, the sum of the absolute values between **x** and the components of the vector $\boldsymbol{W}_i^I$ is calculated in the "Distance Function" to produce A_i, as in Equation 2.10. The dimension of the input weight matrix $\boldsymbol{W}^I$, as shown in Equation 2.11, is QxR. Each row of $\boldsymbol{W}^I$ is referred to as input weight vector associated with a corresponding hidden neuron. Then, the value A_i is multiplied by the bias constant *B* in the "Scaling Function" to provide a_i, which is called the radial distance and shown in

Equation 2.12. The last step is to calculate the radial function outputs $h_i(a_i)$ to provide the neuron's output h_i, which represents the hidden neuron output.

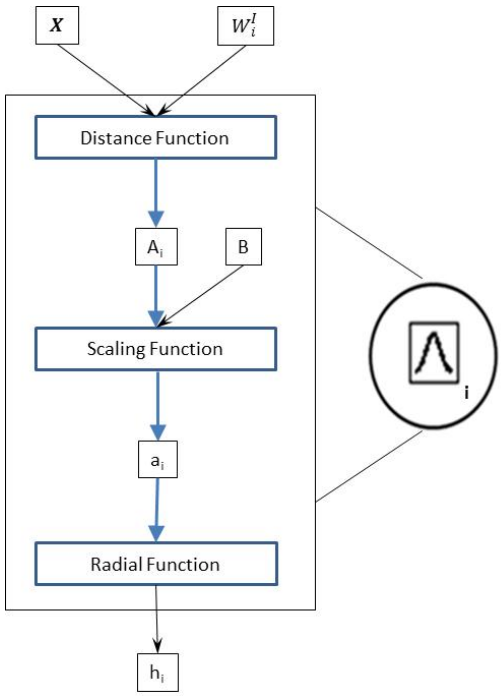

Figure 2.6: The ith neuron at the hidden layer.

In Figure 2.6, $[x_1, x_2, \ldots, x_R]$ represents the network input parameters, $\boldsymbol{W}_i^I$ is the input weight vector of the ith neuron in the hidden layer (dimension= 1xR), A_i is the distance function output (dimension= 1x1), *B* is the bias constant (dimension= 1x1), a_i is the scaling function output (dimension= 1x1), and h_i is the radial function output. The bias *B* is responsible for the network sensitivity, which is directly calculated from a network parameter called the Gaussian width (GW). More detail about those two terms will be provided in the next subsection of this chapter.

$$\boldsymbol{W}_i^I = [w_{i1}^I \;\; w_{i2}^I \ldots . w_{iR}^I \;] \tag{2.9}$$

$$A_i = \sum_{j=1}^{R} \left| w_{ij}^I - x_j \right| \tag{2.10}$$

$$\boldsymbol{W}^{I} = \begin{bmatrix} w_{11}^{I} w_{12}^{I} \dots w_{1R}^{I} \\ w_{21}^{I} w_{22}^{I} \dots w_{2R}^{I} \\ \dots \dots \dots \dots \dots \dots \\ w_{Q1}^{I} w_{Q2}^{I} \dots w_{QR}^{I} \end{bmatrix} \tag{2.11}$$

$$a_i = A_i * B \tag{2.12}$$

$$h_i = rad(a_i) \tag{2.13}$$

As stated, the radial function output h_i is a function of distance, which is presented as a_i in the GRNN hidden neuron. Figure 2.7 shows the general radial function, where the a_i represents the distance n from the center in the figure (as in Equation 2.14). Different types of radial functions are used as a radial transfer function in the RBNN. Buhmann et al. (2003) showed that the Gaussian function is the most popular radial function, thus it will be used in this study. Note that Equations 2.13 and 2.14 are the same; Equation 2.14 is a generalized form of Equation 2.13.

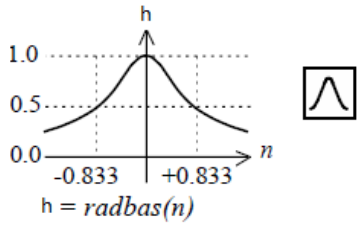

Figure 2.7: Radial function.

$$h(n) = \exp\left(-\frac{(n-c)^2}{r^2}\right) = \exp\left(-\frac{(a_i)^2}{1}\right) = \text{ rad } (a_i) \tag{2.14}$$

where : *n* is the distance from the function's center.

c: the center of radial function (c=0).

r: the radius of the radial function (r=1).

In Figure 2.8, $\boldsymbol{h} = [h_1, h_2, h_3, h_Q]$ represents the output from Q hidden neurons, which are provided to each neuron in the output layer. The figure shows the kth output neuron, where the sum of the dot product between the provided $\boldsymbol{h}$ and the output weight

vector $\boldsymbol{W}_k^O$ is calculated and divided by the sum of $\boldsymbol{h}$ components (as in Equation 2.15). The output y_k essentially represents one of the network's final outputs, where there are N outputs ($\boldsymbol{y} = [y_1, y_2,\ldots, y_N]$), and the output is a weighted sum for the components of the received $\boldsymbol{h}$**.** The dimension of the output weight matrix $\boldsymbol{W}^O$, as shown in Equation 2.16, is NxQ. Each row of $\boldsymbol{W}^O$ is referred to as a weight vector associated with a corresponding output neuron.

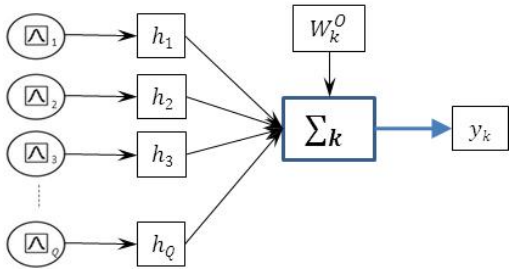

Figure 2.8: The kth neuron at the output layer.

In Figure 2.8, $\boldsymbol{W}_k^O$ represents the output weight vector of the kth neuron in the output layer (dimension= 1xQ), and y_k is the kth network output (dimension= 1x1). Equation 2.18 shows how the general kth network output is represented as a radial function of the input and weight vectors.

$$y_k = \frac{\sum_{q=1}^{Q} h_q . w_{kq}^o}{\sum_{q=1}^{Q} h_q} \tag{2.15}$$

$$\boldsymbol{W}^O = \begin{bmatrix} w_{11}^o w_{12}^o \ldots w_{1Q}^o \\ w_{21}^o w_{22}^o \ldots w_{2Q}^o \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ w_{N1}^o w_{N2}^o \ldots w_{NQ}^o \end{bmatrix} \tag{2.16}$$

$$\boldsymbol{W}_k^O = \begin{bmatrix} w_{k1}^0 & w_{k2}^0 \ldots . w_{kQ}^0 \end{bmatrix} \tag{2.17}$$

$$y_k = \frac{\sum_{q=1}^{Q}\left(exp\left(-\left(\sum_{j=1}^{R}\left|w_{ij}^I - x_j\right|\cdot B\right)^2\right)\cdot w_{kq}^o\right)}{\sum_{q=1}^{Q}\left(exp\left(-\left(\sum_{j=1}^{R}\left|w_{ij}^I - x_j\right|\cdot B\right)^2\right)\right)} \tag{2.18}$$

As mentioned previously, GRNN is a special type of RBNN because it is trained quickly and provides accurate results without having any optimization. The training process is done simply in two steps. First, define the Gaussian width (GW), which is the width of the Gaussian transfer function. Second, setting the values of $\boldsymbol{W}^I$ and $\boldsymbol{W}^O$ to be the inputs and outputs, respectively, of the training cases. Each training case consists of a set of input $\boldsymbol{x}$ and output $\boldsymbol{y}$. For the nth training case, the nth row of $\boldsymbol{W}^I$ takes the input vector $\boldsymbol{x}$, while the nth column of $\boldsymbol{W}^O$ takes the output vector $\boldsymbol{y}$**.** The network has a fixed number of neurons in the hidden and output layers, which are set to Q and N, respectively. Since the radial output from the hidden neuron depends on the distance between $\boldsymbol{x}$ and $\boldsymbol{W}_i^I$ (Equation 2.13), the network provides outputs that are calculated as a weighted sum for the radial outputs of the closed training cases to the new input case, as in Equation 2.15. For example, if the new input vector $\boldsymbol{x}$ equals the weight vector $\boldsymbol{W}_i^I$ (i.e., the same training cases are used as input), the sum of differences between their components will be 0 and leads the radial function's output to be h_i = 1. Then, multiplying h_i by the output weight vector $\boldsymbol{W}_i^I$ produces exactly the component of the weight vector, which are the training outputs for that specific training case.

The remaining question in constructing and training the GRNN is how to define the GW for such a network? Although constructing and training the GRNN is easy, the network does not work well at any GW value. Hence, this parameter is the most critical component when constructing the GRNN because it is the only parameter that needs to be determined by the user and its value affects the network performance. The following section presents the details for the GW and its importance for successful network performance.

2.3.2.2 Network parameter (Gaussian width, GW)

As stated, the GW determines the width of the Gaussian function, which is the transfer function in the hidden neurons. The importance of the GW is that when the

distance A_i equals the GW, the radial output h_i = 0.5 (Equation 2.13), assuming the remaining components in the vector $\boldsymbol{h}$ are zeros. Thus, the network output y_k is dropped to 0.5 of the output training value saved in$\boldsymbol{W}_k^O$. Hence, the GW essentially determines the distance around each training case where it is still able to provide radial output h_i, and eventually its contribution in predicting any input **x**. That is why the GW value determines the network performance and its prediction ability. However, there is no specific mathematical formula to define a proper value for the GW to make the network perform in the best way.

To understand the exact mathematical meaning for the GW, let us take two different Gaussian functions with different GW values (GW1 and GW2), shown in Figure 2.9. In the figure, the function with GW1 is narrower than that with GW2, and covers less space in the shown two-dimensional plot. Now, suppose the calculated scaling function a_i (Equation 2.12) at the hidden layer equals 4 (a_i= n= 4). The radial output for this value (n=4) for the function with GW1 is almost 0, while it is almost 0.4 for the function with GW2.

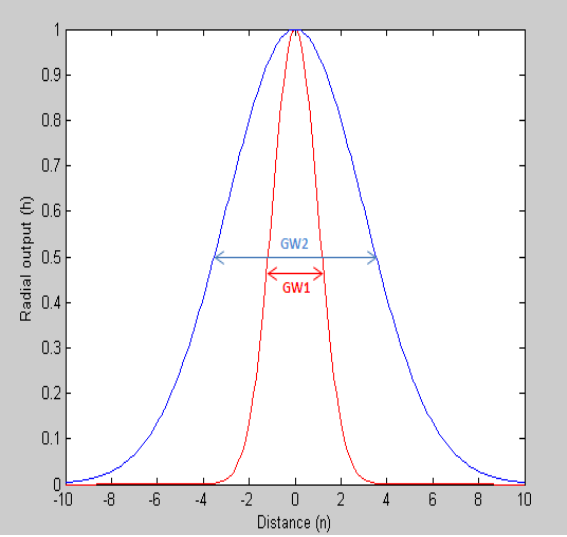

Figure 2.9: Two Gaussian functions with different GWs.

Now, which value between GW1 and GW2 provides more accurate results for the network at n equals 4? GW1 is good to be used as long as n is within the domain of another Gaussian function in the training space (i.e., another training case $\boldsymbol{W}_i^I$ has non-zero output for n=4). So, the prediction becomes more accurate than that with GW2, which in this case turns out to be overlapping with another training case in predicting the same point. On the other hand, if n is not within the domain of any other point in the training space, the function with GW2 is better to be used because the network with GW1 provides output equals zero. For such a case where the network output is zero, this distance or space is called a dark or empty space because the network cannot predict outputs for any point in that space. It has been impossible until now to calculate the optimal GW for a specific application (Specht et al., 1991). Moody et al. (1989) proposed that the GW could be determined heuristically after normalizing the input data by testing GW values near the midpoint of input values. This, again, presents the importance of inputs normalization in the training process.

The above discussion indicates that the proper GW is a trade-off between covering most or all grid space and achieving acceptably accurate prediction. Having narrow GW value could leave some dark spaces in the training space. Thus, the network is less efficient in predicting all inputs within the training space. On the other hand, larger GW decreases the accuracy of the predicted output because many neurons (i.e., training cases) are overlapping and firing for the same input. Firing means that the output of the neuron is not zero. In this case, any output is always a result of contributions from many neurons in high fractions, which decreases the accuracy of predicting a specific point in the grid space. Hence, this parameter needs to be chosen depending on the application so that the best network performance is achieved (Specht et al., 1991).

In the context of the presented equations for the GRNN layers, the GW effect on the network performance is introduced through the bias *B*. The GW determines the bias *B* in Equation 2.19, and *B* shows that effect on the network in the scaling function

(Equation 2.12). From the radial function shown in Equation 2.14, the function $h(n)$ decreases to 0.5 at distance *n* equals to 0.833 from its origin. This value (0.833) is obtained in Equation 2.20 by solving the values in Equation 2.14. So, this parameter (*B*) is affected and changed by changing the GW of the function and provides another way to control the overall network performance. These calculations are from Wasserman et al. (1993). By defining *B* in terms of the GW, we can control the generated network sensitivity and have more efficient performance.

$$B = 0.833/GW \tag{2.19}$$

$$\ln\left(\exp\left(-\frac{(n)^2}{1}\right)\right) = \ln(0.5) \rightarrow (n)^2 = -\ln(0.5) = \ln\left(\frac{1}{0.5}\right) \rightarrow n = \sqrt{\ln(\frac{1}{0.5})} = 0.833 \tag{2.20}$$

CHAPTER III

NEW METHODOLOGIES FOR ARTIFICIAL NEURAL NETWORK (ANN) WITH HUMAN MODELING

Implementing an artificial neural network (ANN) to solve digital human modeling (DHM) problems requires solid construction of the network. Constructing the network includes three main steps: 1) defining the problem input and output sets (training cases), 2) training the network using the training sets to simulate the targeted problem or task, and 3) testing the network performance before considering it for final use. This chapter presents new methodologies for automatically constructing the network with best performance. There are many types of ANN that are used in various system predictions. Thus, this research also analyzes and refers to the best candidate network to solve DHM problems. The chapter provides the following contributions:

1. Selecting the proper type of ANNs to study DHM problems. General regression neural network (GRNN) is found to have advantages for handling DHM problems.
2. Developing a semi-automated training and testing (construction) process. With the newly developed strategy, the network is automatically trained using any set of cases. Then, the performance of the built network is tested before it is used.
3. Introducing a new automatic strategy for determining the Gaussian width (GW) parameter for optimal network performance for any DHM problem.
4. Introducing task-based definition (i.e., task formulation) for the DHM applications using ANN.

3.1. Selection of Network Type

As mentioned in Chapter 2, for solving regression problems like DHM applications, there are two major types of ANNs: feed forward (FFNN) and radial basis (RBNN) networks. GRNN belongs to the RBNN type of networks, where the outputs

depend on the radial distance between the inputs and network weights inside the neurons in the hidden layer. Note that, given the many available variations of ANNs, selecting the appropriate form and associated parameters for a particular application can be more of an art than a science. Thus, one of the contributions of this work is the determination of the appropriate ANN and ANN parameters for application to DHM predictions. Many types of ANNs were used in the literature to solve and study various DHM problems, but applications of the GRNN to these problems are still limited.

In this research, some initial work was done in comparing the predicted results from FFNN and RBNN for the task of human motion, which is discussed in detail in Chapter 4. The RBNN results were better than the FFNN results, because FFNN has memory and convergence problems when applied to problems with a large number of outputs. Human motion prediction and most DHM problems have a large number of outputs.

Then, the accuracy of results produced from different types of RBNN was tested, and GRNN was found to have the most accurate results. In addition, by investigating the general advantages and disadvantages of various ANNs, GRNN was found to be the most appropriate for solving DHM problems. Therefore, the GRNN is the network chosen to be applied in this thesis. In general, using GRNN has the following advantages:

1. Constructing and training quickly without memory or training time problems.
2. Smoothing out the regression curve between the training grid points. The GRNN can predict any off-grid point by finding the weighted sum of the closest grid points to that point. Thus, it is able to predict the system behavior accurately using a small number of training cases.
3. Providing realistic results without converging to a poor solution, because there is no iterative optimization in the GRNN construction. Poor convergence is one of the main limits when using FFNN and leads to unexpected results for some inputs.

4. Having a smaller number of heuristic network parameters to be determined in advance than any other type of RBNN.

3.2. Semi-automated Network Construction Process

When ANN is used for any application, the training process is the most challenging part of the network construction. In general, the training process starts with collecting the training cases in two large matrices. One matrix has the inputs, and the other the outputs. Then, the network parameters are set up, especially GW. Finally, the network is created and trained to predict the output that corresponds to the input.

In anticipation of other users and potential integration within the Santos software (see Appendix A), training process steps are performed in this thesis using a proposed semi-automatic method. Testing the performance of the constructed network is also included in the proposed method. This method makes use of GRNN in DHM applications simple enough to be carried out by a person who is not familiar with the ANN development process. Figure 3.1 shows a schematic diagram for this new network construction process.

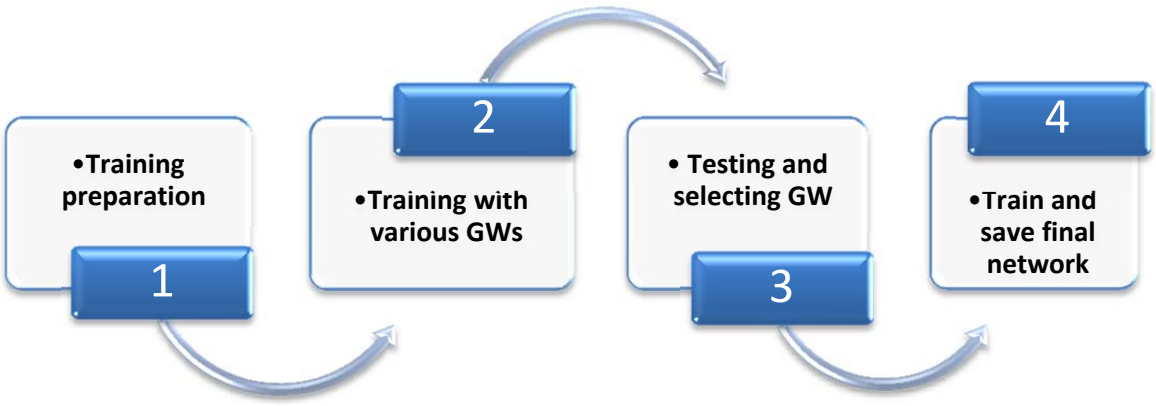

Figure 3.1: Schematic diagram of the general automated steps in constructing the GRNN.

The four steps in the above figure are fully automated. The only manual part of the method is organizing the inputs of all training cases in one file for the task to be trained. For each training case, the values of the input parameters should occupy one column in the file. The entire process takes between 1 and 2 minutes.

3.2.1 Training preparation

As mentioned earlier, before the training starts, all training cases are combined in two matrices, one for each of the inputs and outputs. The inputs, in specific, are then normalized. Input normalization is done when using any type of ANN and for any application. A hyperbolic tangent sigmoid function (TSF) is used for the normalization in this research because it is the standard normalization function in ANN, as shown in Chapter 2. Figure 3.2 shows the TSF, which compresses the inputs to all be between -1 and 1 (Vogl et al., 1988).The mathematical formulation for the function is shown in Equation 3.1.

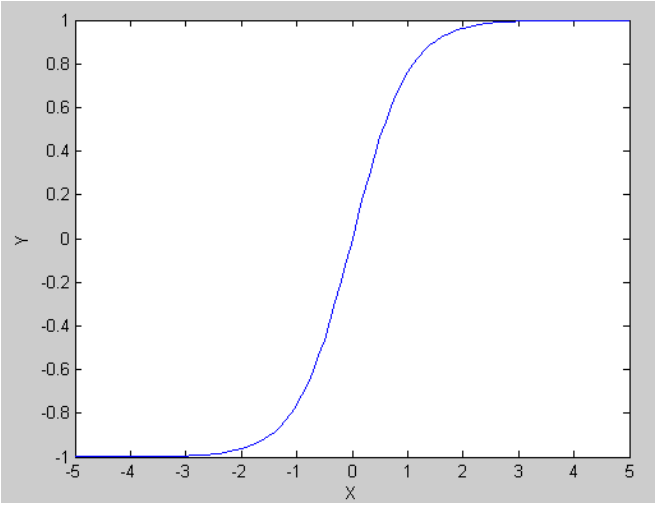

Figure 3.2: The hyperbolic tangent sigmoid function.

$$y = \frac{e^{2x}-1}{e^{2x}+1} \tag{3.1}$$

Input normalization is important for proper network performance because it confines all the inputs from different types and ranges to between -1 and 1. Therefore, variation of the inputs is lower, which allows the network to be more stable in predicting and handling the outputs of different provided input values. In addition, determination of the network parameter (GW) becomes easier because of the uniform range of different inputs or features, as shown in Chapter 2. For example, the walking task, which is described in Chapter 4, has inputs that include velocity (with values between 0.8 and 1.6) and backpack weight (with values between 0 and 350). If the network is trained on these input ranges, the network sensitivity for the velocity and backpack changes will not be consistent. In other words, the network will not predict the changes well, because the ranges of the inputs are not similar. Therefore, once the normalization is completed, both velocity and backpack weights will have a range between -1 and 1. Then, the network handles the inputs better and provides more accurate prediction for any input change.

3.2.2 New strategy for Gaussian width (GW) selection

After training preparation is completed, it is followed by the training and testing stages of the network construction. This section discusses steps 2 and 3 of the constructing process illustrated in Figure 3.1. In addition, a new strategy is presented for selecting GW that improves the network performance. The strategy is embedded in the training and testing stages.

As mentioned in Chapter 2, radial function output decreases to 0.5 when the difference between the inputs and neuron weight vector equals GW value. The inputs are all normalized between -1 and 1, so the maximum difference is 2. Therefore, the initial guess for the GW value is to be somewhere around the middle of inputs range. The inputs range is 2, which is the absolute difference between -1 and 1.

When using GRNN, the user should define the GW for the network. Since there is no specific formula to calculate the best GW value for such task, many studies have

found GW by trial and error. This thesis proposes a heuristic strategy for automatically determining GW, as shown in Figure 3.3.

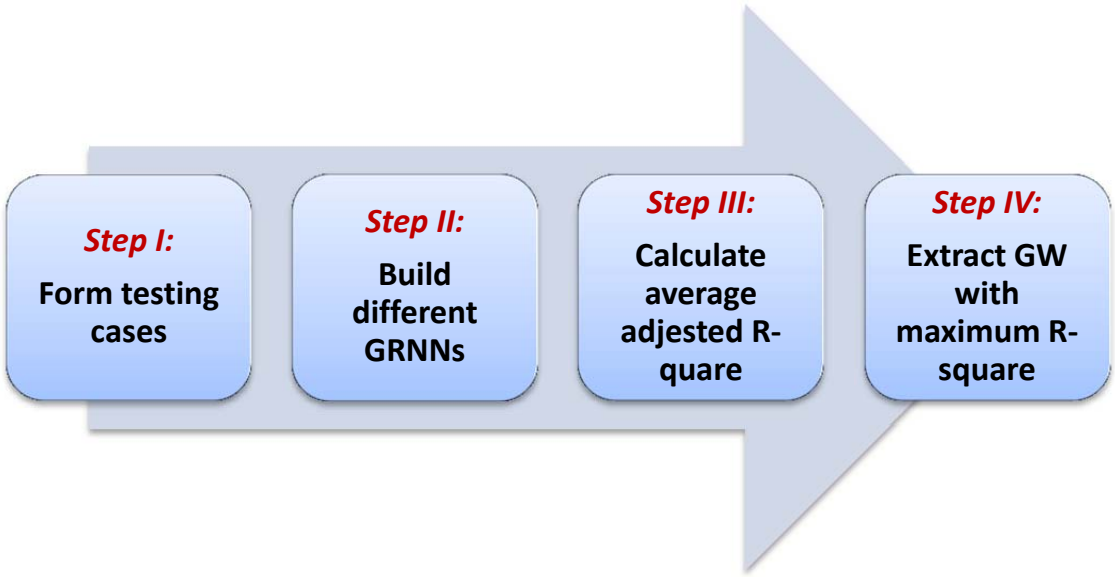

Figure 3.3: Steps for automatically determining the GW of GRNN.

In ***step 1***, the collected normalized training data are separated into two parts, as shown in Figure 3.4. The first part is called the true training data and includes all training cases except three randomly selected cases. These cases are the cases that are used to train the network. The second part has the three excluded cases, and it is called test data. The randomly selected cases are chosen not to be one of the extreme training cases. Extreme cases are those that have inputs with the maximum or minimum training values (e.g., velocity equals 0.8 or 1.6 in the walking task). In other words, test data should not include any of the cases located on the boundary (corners) of the training grid (the grid space). That is because the GRNN is unable to extrapolate and predict any case outside its grid space. The network does not have a grid point for them, and they are considered

as new inputs for the network. Hence, they are used to test whether the network is trained well.

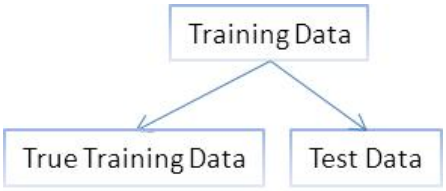

Figure 3.4: Splitting collected training data.

In ***step 2***, 40 GRNNs are created or built, one for each of 40 different GWs. The GW values range from 0.05 to 2 in increments of 0.05, where this range represents the all possible GW values within the inputs range. The selected increment is small enough to exactly follow and specify the most accurate GW. Larger increments might pass the proper GW, while lower increments are useless because they are too small to have a notable effect on the produced network. The GW should also be positive.

In ***step 3***, for each created network, adjusted R-square values are calculated for the three testing cases. In addition, three other randomly selected on-grid cases (training cases) are evaluated by calculating their adjusted R-square values. The adjusted R-square value represents the degrees of accuracy between the predicted results from the created GRNN and the exact results (the collected results from the training source). Including the chosen three training cases in the testing phase is important for generalizing the prediction ability for the network. If the network is evaluated using only the three selected testing cases, its prediction might be good for those cases only. So, the trade-off between on-grid (training cases) and off-grid (testing cases) will be the best evaluation for the candidate network. Then, for each created network, the average R-square value is

calculated for the six chosen cases (the testing and training cases). This step is considered the test phase in the general network construction process.

After all created networks are evaluated, the average R-square values that resulted in step 3 are compared. So, there are 40 averages of adjusted R-square values, each produced from the results of one network with GW value differs from the rest networks. Then, ***step 4*** entails choosing the maximum value in the vector, which corresponds to the best GW. By the end of this step, the training and testing steps are finished, and the best GW value for proper network performance has been identified.

The walking forward task, which is described in detail in Chapter 4, is used as an example to show how to apply the new proposed methodology. For that task, there are 12 inputs, 390 outputs, and 55 collected training cases. Each training case is represented by a combination of inputs and their corresponding outputs. The training cases are first split into 52 true training cases and 3 for testing cases. Then, using the 52 training cases, 40 GRNNs are built with GW values ranging from 0.05 to 2, in increments of 0.05. For each network, six adjusted R-square values are calculated for the three testing cases and three training cases (the cases were case 7, 23, and 41 in Table B.1). So, six cases are set to evaluate the performance of each created network, three from test cases and three from the training cases. The calculated R-square differs from one network to another, because each network has a different GW value. Figure 3.5 shows three adjusted R-square values resulting from three different networks for the same testing case. In the figure, the exact joint torque values from the training source (PD-Torque) are on the vertical axes, while the predicted joint torque values (GRNN-Torque) are on the horizontal axes. The shown R-square values are from three networks that were built using different GWs. The R-square is obtained by plotting the predicted outputs from the network versus the exact outputs from the training source. Larger R-square values mean more accurate results (i.e., closer to the accuracy line). Thus, the figure shows that the second network, the network with GW=0.45, has the highest accurate result (highest R-square value) in predicting this

case. However, for such network to be the best in predicting a problem, the network should have the highest R-square values in predicting all testing cases.

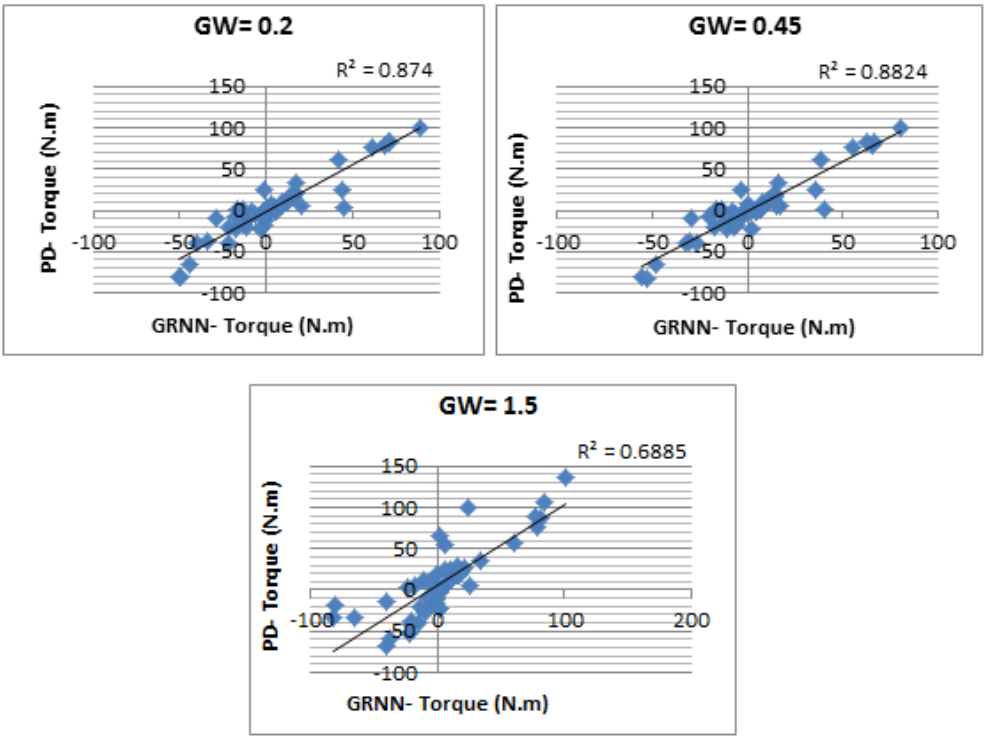

Figure 3.5: Three adjusted R-square plots for the same testing case resulting from three different GRNN networks; the networks differ in the used GW value.

Next, the average of the six resultant R-square values is calculated. There are 40 resulting average R-square values, which present the accuracy achieved form the 40 created networks. For the walking task, the highest average value was 0.87, which was for the network with GW equal to 0.45.

3.2.3 Training and saving the final network

After the GW value for best GRNN performance is assigned, this GW is used to create and save the final network. In this stage, only one GRNN is created using the

extracted best GW (i.e., the network with highest testing results) and trained using the same true training cases. By completing this step, the automatic network construction process is finished, and the network is ready to predict any new inputs for the trained task.

3.3. Task-based Digital Human Modeling (DHM) Applications Using Artificial Neural Network (ANN)

As mentioned in the literature review, ANN is used in various applications, including those in the DHM field. While there are some applications for directly using ANN in posture prediction, it is not used for predicting full human motion in a task-based manner that has many inputs. For example, this thesis presents the prediction of a walking forward task, detailed in Chapter 4, where the task inputs include the step size as backpack weight. The applications in posture prediction and motion prediction also differ depending on the task to be performed. Hence, this thesis presents a successful use of GRNN in predicting various task-based DHM applications. The following provides a brief description of the successful use of ANN in this thesis to predict these various DHM applications.

In Chapter 4, two different human motion prediction applications are predicted using GRNN. In motion prediction, the main and biggest problem is predicting all DOFs, which is a relatively large number, accurately. By using the GRNN, the proposed training strategy, the motion prediction is achieved in a highly accurate and realistic manner. Details about these applications will be presented in that chapter.

Chapter 5 presents two other applications for using GRNN to predict human postures. This type of ANN has never been used in posture prediction problems or in the context of a large number of DOF. This application obtained some useful conclusions from that application and potential limitations on the current use of ANN in posture prediction problems. This chapter has more details about the applied tasks.

The third application, which is presented in Chapter 6, includes using GRNN to predict the weights for human PMs within the context of multi-objective optimization problems that control human posture prediction. Those PMs have not yet been fully studied; no one has studied the proper PM combinations (i.e., PM weights) for best posture prediction results. This thesis provides initial work on predicting the weights of PMs to make sure this could be a potential area for future research. This could lead to extracting and understanding the specific correlations between the PMs. In addition, new methodologies for extracting the weights of human PMs that contribute in posture prediction, which will be described later, are proposed where it shows strong potential for understanding what drives the human when performing different tasks.

CHAPTER IV

NEURAL-NETWORK-BASED MOTION PREDICTION

4.1. Introduction

The majority of digital human modeling (DHM) applications in industry and academia require human interaction and dynamic simulation. Studying human dynamics or motion in specific is also critical for injury prevention and for better understanding human behavior. As mentioned in Chapter 1, predicting human motion is still immature in terms of computational speed and realistic behavior. In reality, human motion has many constraints and factors that make any developed model slow to produce or predict the motion. In addition, the lack of standard motion strategy also makes the prediction of realistic motion hard because people behave differently.

However, there is new promise for embedding an artificial neural network (ANN) in direct prediction of human motion. This capability allows one not only to predict human behavior but to study more effectively how people behave the way they do. Although ANNs have been studied extensively and are currently applied to a variety of problems, their benefits have not yet been realized in the context of human-motion models. In this context, we find ANN to be computationally fast and to provide new insight into which articulated degrees of freedom (DOFs) play the most significant role in defining one's motion.

In general, although many have used ANNs to study specific aspects of human motion, the current state of the art does not demonstrate the use of ANNs for direct manipulation of joint angles in the context of a complete human model with a large number of DOFs. Thus, we propose not only exploring the use of ANNs for predicting whole-body motion but doing so in the context of a complete 55-DOF DHM. Furthermore, we demonstrate the ability to modify task parameters, such as range of motion and applied load. ANNs are coupled with optimization-based dynamic motion

prediction and thus alleviate the need for pre-recorded data. However, the proposed approach can be used with experimental data as well, should it be available.

In this chapter, general regression neural network (GRNN) is used to predict motion for two tasks: 1) walking forward with a backpack and 2) jumping up on a box. These tasks are performed to examine ANN's ability to predict human performance motion with different input parameters depending on the task. For both tasks, this chapter discusses: 1) definition of the task, training process, and network properties, 2) results compared statistically (objectively) and visually (subjectively), and 3) conclusions and current limitations. This chapter includes the following novel contributions:

1. Implementing the proposed methodologies in Chapter 3 to predict two task-based motion predictions using GRNN. Two different tasks are defined and predicted properly.
2. Demonstrating the ability to modify task parameters and see the results in real time. Every time the task inputs change, predictive dynamics (PD) takes between seconds to minutes to predict the outputs. Using and training the GRNN to predict the task provides immediate (real-time) outputs for any change in the inputs.
3. Predicting a relatively large number of outputs of various forms: joint angles, joint torques, and ground reaction forces (GRFs). Thus, demonstrating the advantages of using GRNN compared to other types of ANN like feed forward neural network (FFNN).
4. Coupling ANN with predictive dynamics to provide a faster predictive system.
5. Developing an automatic algorithm to collect training cases for any motion task. The algorithm changes all required input files each time before running the PD, and saves the results (new training case) under a new name in a consecutive manner.
6. Incorporating joint torques and GRFs as part of the predicted outputs beside joint angle control points.

4.2. Proposed Task Definition, Training Process, and Network Properties

Task-based motion prediction means that motion is predicted depending on the task to be accomplished. Both tasks (walking forward and jumping up on a box) are defined separately in this section to specify the inputs and outputs of each task. Task inputs are called task parameters and vary according to user inputs to the predicted motion by the network. After definition of the task inputs and outputs is complete, the new proposed semi-automatic training and testing method, shown in Chapter 3, is applied to construct the best network for the specified task. During the training and testing processes, the network properties are defined and Gaussian width (GW) is determined to provide the maximum network performance.

Training the network can be done using experimental data. However, gathering such data can be time consuming and costly, especially when considering different tasks and variations in subject anthropometry. In addition, some data, like joint torque, simply may not be available. Thus, we propose using PD to train the ANN. The PD algorithm formulation is described for each task separately.

As a result, many training cases for different input combinations are generated to train the network at different input values or conditions. The work of collecting the training cases and training the network was done on a Windows 7 computer with an Intel ® Core™ 2 processor and 8 GB of RAM. PD takes between 5 to 20 minutes to predict and generate each training case. The speed of computation is one of the issues that the GRNN could solve, if it works well.

4.2.1 Walking forward

The first motion task is walking forward with a backpack. This task is proposed for the following reasons: 1) to examine the general GRNN's ability to predict a task with a relatively large number of training cases, inputs, and outputs, 2) to examine the

GRNN's ability to predict two different types of outputs, 3) because it is one of the most common tasks when studying human motions, and 4) because the walking task is one of the mature tasks that are predicted by PD and validated by motion capture.

The walking forward task has many parameters to be changed as network inputs. Before using GRNN to predict this task, initial work was done using FFNN to predict the same task. The FFNN did not work well for this task because it experienced memory problems when handling the large number of outputs and training cases. The network worked only when the number of outputs and training cases were reduced to a smaller number (around 50 outputs). Even with the smaller numbers, the results from this network were not totally accurate. Thus, RGNN is used to predict the walking forward task and the tasks in the following applications. As stated, the GRNN is used in this task to provide an example of its ability to handle complex DHM problems, like motion prediction, with a relatively large number of inputs and outputs.

The variables that are considered and defined as inputs for the walking task include: 1) motion velocity, 2) backpack weight, 3) four lower-body link lengths (spine to hip, hip to knee, knee to ankle, and ankle to football), and 4) three body joint range of motions (ROMs) (their upper and lower limits). ROMs include the upper and lower limits for the hip, knee, and ankle, each at flexion-extension. Those specific joint ROMs are used because changing their limits has significant effects on the resulting walking task. Table 4.1 shows the input parameters for the training cases for this task. In the table, each of these parameters takes only one of the three listed values as a training value, except the velocity, which has only two values. Each training case uses a combination of the inputs (training inputs) where each input parameter could have any of the three values for each training case. If one or more of the input parameters has/have input value(s) other than the three listed values, the input case is considered an off-grid case or point (off the training grid or points).

Table 4.1: Input parameters for the walking-forward task.

Input parameter	Value 1	Value 2	Value 3
Velocity (m/s)	0.8	--	1.6
Backpack weight (N)	0	175	315
Link1 (Spine to Hip) (cm)	7.8	8.8	9.8
Link2 (Hip to Knee) (cm)	43.5	44.5	45.6
Link3 (Knee to Ankle) (cm)	39.5	42.4	45.4
Link4 (Ankle to Football) (cm)	11.3	11.7	12.1
Joint1 (Hip)- lower limit (degrees)	-123.3	-105	-90
Joint1 (Hip)- upper limit (degrees)	8.7	5	2
Joint2 (Knee)- lower limit (degrees)	5	10	20
Joint2 (Knee)- upper limit (degrees)	149.7	130	110
Joint3 (Ankle)- lower limit (degrees)	7.3	15	20
Joint3 (Ankle)- upper limit	71.6	60	50

In the table, the velocity represents the speed of walking for Santos and is measured in m/sec; the values in the table are the maximum and minimum speeds that an average person could walk. Training values use different weights for the backpack, but the results will not show any backpack visually. The effect of the backpack, however, will be shown on Santos's vertebrae by the degree of bending. Value 3 for backpack weight in the table presents the maximum weight that an average person could carry. The three values of the link lengths are chosen to represent the average length for males, the average length for females, and the average length between the first two values. Lastly, the values of the joint angles are chosen so the first value represents the average male's joint angle limits, both upper and lower. The second and third values were chosen randomly to narrow the ROMs in regular bases. As shown in the table, the second value

has ROM values less than the first value. The third value has narrower limits (less value) than the first two values.

The network's outputs include Santos's joint-angle profiles (joint control points) for 55 DOFs and joint torques. For joint control points, there are six control points that have joint values at different times over the task. Given six control points for each DOF, there are 330 output values for joint angles. In addition, joint torques are considered for the six lower-joint DOFs (three for the hip, one for the knee, and two for the ankle), because they are the most highly articulated during the walking task. Assuming symmetry, the joint torques are evaluated at ten time steps during the walking task. This results in 60 additional output values. RGNN essentially provides a relationship between the inputs and the outputs, which can be called and evaluated quickly. The task is defined with 12 inputs and 390 outputs in total.

After the task was defined, training cases are collected by having different input combinations. There are 52 training cases in which all the used inputs are represented by one of the three values shown in the table above. See Table B.1 (in Appendix B) for all input combinations used in all training cases. The PD mathematical formulations for this task are presented by Xiang et al. (2008). The conceptual formulation for this task is given in the optimization problem that shown in Equation 4.1:

Find: joint angle profiles (control points) (4.1)

To minimize: sum of joint torque-squared

Subject to: 1) joint angle limits, 2) torque limits, 3) equation of motion, 4) dynamic stability, 5) foot strike position, 6) arm leg coupling, 7) self-avoidance, 8) ground penetration, 9) symmetry/continuity condition, and 10) ground clearance.

In motion tasks like walking, there is a large number of training cases. Many files related to the inputs should be also changed in the PD folder before running each training case (i.e., changing some files to set the new input values before running the algorithm). Hence, an algorithm was developed for collecting the training cases automatically and

changing the required files. The algorithm consists of three steps: 1) looping over to collect the training cases, 2) changing all required files to set the new input values each time before running the PD for the next training case, and 3) saving the result file (new training case) under a new name in a consecutive manner.

After the training process is completed, the new heuristic strategy for determining GW found that the best network performance at GW equals 0.45. Using this GW provides the best results for both on- and off-grid points (cases). To summarize, Table 4.2 presents the general definition for the walking task definition as well as the constructed network properties.

Table 4.2: The walking task definition and constructed network properties.

Task Definition	- Walking forward task with backpack - Task variables: • Velocity • Backpack weight • Link length (spine- hip, hip- knee, knee- ankle, and ankle- football) • Joint ROMs (upper and lower limits) ➢ Hip flexion extension: Lower (-123.3, -105, and -90); Upper (8.7,5, and 2) ➢ Knee flexion extension: Lower (5, 10,and 20); Upper (149.7, 130,and 110) ➢ Ankle flexion extension: Lower (7.3, 15,and 20); Upper (71.6, 60,and 50)
Network properties	- Inputs: 12 inputs - Outputs: 390 (Joint control points & torques) • joints control points (55*6)=330 • Joint torques (6*10)=60 - Training cases: 52 cases - Gaussian width= 0.45

4.2.2 Jumping up on box

The second task that this study tests for using GRNN in motion prediction is jumping on a box, shown in Figure 4.1. This task is proposed for the following reasons: 1) to test GRNN's ability to predict a task in which feet and hands should be located (reach) at specific places (i.e., addressing the contact points), 2) because jumping up on a

box is a complicated task that has many constraints, and it requires highly accurate results to be predicted, and 3) to predict ground reaction forces (GRFs) at both feet along with joint angle control points as outputs from the same network. Joint torque values are not included in this task because we need to study the network prediction ability for different types of outputs along with the joint angle control points. Predicting GRFs also is more important than joint torques for this task.

Figure 4.1: Task of jumping up on a box.

Box height is the main variable for this task because it has the most effect on the resulting task when it changes. Santos successfully shows the cause and effect when the height changes from 0.5 to 1 meter. He visually shows the changes by jumping higher when the box height is 1 meter. The PD fails to provide a feasible solution when box height exceeds 1 meter. Therefore, the box height limits for the training are 0.5 to 1 meter. The ROMs are not included in this task because the range of variation in ROM that resulted in feasible solutions was rather limited. There are less feasible solutions for this task than walking because there is one more function to be minimized in this task; the task formulation will be described later in this section. Thus, this task has the following

input parameters: 1) box height and 2) four link lengths. Table 4.3 shows the five input parameters with their training values for the jumping up on a box task.

Table 4.3: Input parameters and training values for the task of jumping up on a box.

Input parameter	Minimum	Maximum
Box height (cm)	50	100
Link1 (Spine to Hip) (cm)	7.8	9
Link2 (Hip to Knee) (cm)	38	43
Link3 (Knee to Ankle) (cm)	39	39
Link4 (Ankle to Football) (cm)	9	12

The links in this task are the same links that were considered for the first task: 1) spine-hip, 2) hip-knee, 3) knee-ankle, and 4) ankle-football. The table also presents the input values that are used in the training cases. During training case collection, the box height value in the current case is a 5 cm increment from the previous one. The link lengths, however, only have two fixed training values. The minimum values are the only values used with the first set of training cases, where the box height is the only changeable value (collecting 11 cases). Then, the maximum values are used and fixed with repeating the same box heights to collect another set of training cases (another 11 cases). There was no collected training case created using mixed link lengths from maximum and minimum values.

Similarly to the walking task, jumping up on box task has two types of outputs in the collected training cases: joint splines (joint angle control points) and GRF, for both feet, whose maximum or peak values are the most important. PD provides a file for GRF with hundreds of values that are tracked over the task time and for both feet. Figure 4.2

shows the typical GRF plots, for both feet, that are produced from the PD at some training case. In each plot, there are hundreds of sampling points which are represented as points. Each sampling point has a force value that shows the GRF at the foot at specific time.

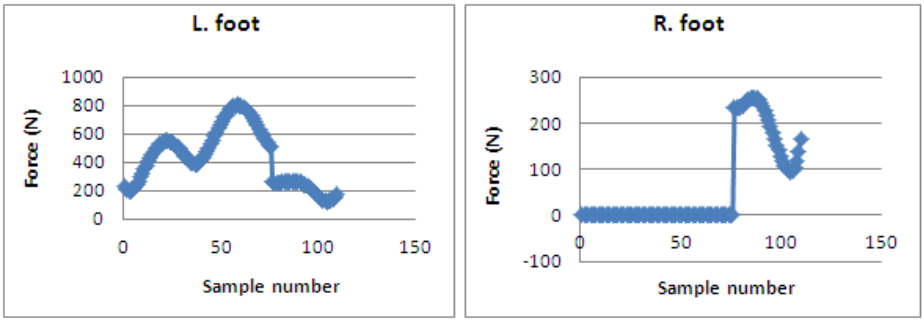

Figure 4.2: The typical GRF plots, for both feet, at some training case.

Most of the time, researchers are interested in the maximum or peak segment or region of the GRF at each foot. In addition, the right foot has zero GRF values during most of the task time, because it is free from any contact during most of the task time. Thus, to decrease the number of outputs, the maximum 20 GRF samples for each foot are extracted from the training files, and the network is trained to predict them as part of the outputs. Figure 4.3 shows the plots of the maximum 20 samples for the same training case that presented in Figure 4.2. The vertical axis represents the force values in (N), while the horizontal axis is for the sample number. The predicted samples are always arranged so that sample number 10 has the maximum value (the peak GRF value). The plots have different scales because the GRF values on both feet are sharply different. In summary, the total number of outputs is 370, 330 for joint splines (joint angle control points) and 40 for GRFs.

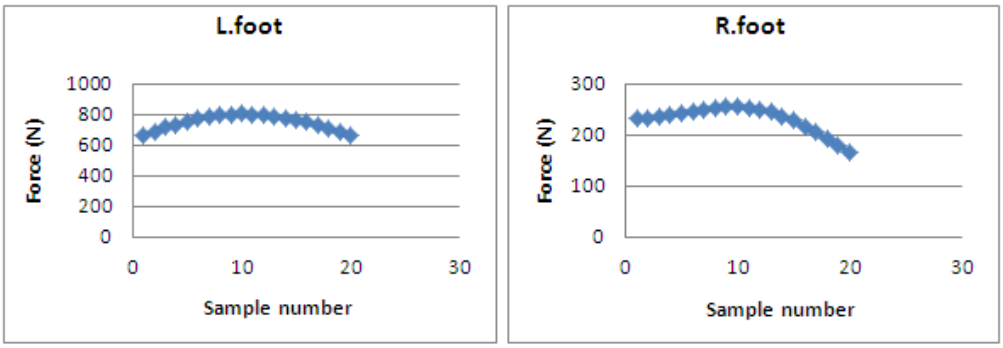

Figure 4.3: The plots of the maximum 20 GRF samples, for both feet, at some training case.

In this task, the left foot is the foot that stays on the ground when Santos is jumping up. So, it carries Santos's weight and the extra force from pushing himself up. This foot has greater GRFs than the right one, which is free at most of the task time. From the above figure, the peak GRF for the left foot is around 800 N, while that for the right foot is around 280 N.

Now, the task definition is completed by having specific known inputs and outputs. In addition, training cases are collected for different on-grid points using the PD algorithm. The next step is to create and train the network, which is semi-automated (see Chapter 3). The training process is done to create a GRNN with 5 inputs, 370 outputs, and 22 training cases. See Table B.2 (in Appendix B) for all input combinations used in all training cases. The conceptual formulation for this task is given by the following optimization problem:

Find: joint angle profiles (control points) (4.2)

To minimize: sum of joint torque-squared+ motion capture

Subject to: 1) joint angle limits, 2) torque limits, 3) equation of motion, 4) dynamic stability, 5) foot strike position, 6) self-avoidance, and 7) ground penetration.

After the automated training process was completed, the network had GW equal to 0.05, which provided the best achievable output prediction from the network. The GW

value in this task is much lower than in the walking task because the jumping up on a box task has fewer input parameters. Thus, the resultant grid space is much smaller for the jumping task, which needs smaller GW to cover all the spaces between the grid points (training points). Another reason for the smaller spaces or gaps is that the increment between input values for the same parameter in the jumping task is smaller than that in the walking task. So, the grid points are closer to each other in the jumping task. To summarize this task, Table 4.4 presents the general definition for the jumping up on a box task definition as well as the constructed network.

Table 4.4: The jumping up on box task definition and constructed network properties.

Task Definition	- Jumping up on box - Task variables: • Box height • Link length (spine- hip, hip- knee, knee- ankle, and ankle- football)
Network properties	- Inputs: 5 inputs - Outputs: 370 (Joint control points & Ground reaction forces) • joints control points (55*6)=330 • Ground reaction forces (2*20)=40 - Training cases: 22 cases - Gaussian width= 0.05

4.3. Results

This section presents the predicted outputs from the GRNNs that were used in both tasks. These outputs are presented, evaluated, and compared with those exact outputs from the PD algorithm that were used to train the networks. Both tasks are discussed separately with relatively similar comparison strategies, and the results are also divided and shown in subsections for each task. Both subjective and objective results are evaluated and presented where both the visual and statistical results should be accepted in motion prediction tasks. The walking forward task has two types of outputs (joint angle

profiles and joint torques), which are evaluated separately. The jumping up on a box task also has two types of outputs (joint angle profiles and GRFs), which are also presented in the same way the first task is shown.

4.3.1 Walking forward

This part evaluates and analyzes some testing cases on the predicted walking task from the trained GRNN. Those cases include cases with on- and off-grid input points. The network was trained to predict the on-grid cases but not the off-grid. Six cases are tested and evaluated for this task: three on-grid cases and three off-grid cases. The three on-grid testing cases are shown in Table 4.5.

Table 4.5: Walking forward input parameters for three on-grid testing cases.

Input parameter	Case 1	Case 2	Case 3
Velocity (m/s)	0.8	1.6	1.6
Backpack weight (N)	0	315	315
Link1 (Spine to Hip) (cm)	7.8	8.8	8.8
Link2 (Hip to Knee) (cm)	43.5	44.5	44.5
Link3 (Knee to Ankle) (cm)	39.5	42.4	42.4
Link4 (Ankle to Football) (cm)	12.1	11.7	11.7
Joint1 (Hip)- lower limit (degrees)	-123.3	-123.3	-123.3
Joint1 (Hip)- upper limit (degrees)	8.7	8.7	8.7
Joint2 (Knee)-lower limit (degrees)	5	5	20
Joint2 (Knee)-upper limit(degrees)	149.7	149.7	110
Joint3 (Ankle)-lower limit(degrees)	7.3	7.3	15
Joint3 (Ankle)- upper limit	71.6	71.6	60

As noticed, the input values for the cases in the above table are same for those used in the training cases in Table 4.1. Those three testing cases were used in the training process, and they are randomly chosen from the training cases. These cases are used for testing and comparing the network ability to predict any point after the training is complete. The accuracy of predicting these cases should be the same for the remaining untested training cases, because the network has the same behavior for predicting on-grid cases. If the network predicts the chosen cases well, predicting the other training cases will have the same degree of accuracy. Comparison is performed on each one of these cases between the predicted outputs from the network (testing outputs) and the exact results from the PD (training outputs).

Other results that are shown in this study include three off-grid testing cases. Generally, studying those cases is more critical than on-grid ones, because they are never used to train the network and they test the general performance of the network for future prediction. Off-grid points always have less accurate results than the on-grid ones. Those testing cases are presented in Table 4.6, where their input values are different from those used to train the network. These cases are chosen to cover various input combinations. From the table, Case 1 has relatively the furthest input values from the training values. Case 2 and Case 3 also present other input combinations with different input values. The resultant accuracy for predicting these cases should be similar for any other off-grid cases. Comparisons for all testing cases, both on- and off-grid cases, are performed between the predicted outputs (results) from the GRNN and exact outputs (results) from the PD.

Table 4.6: Input variables for three off-grid testing cases.

Input parameter	Case 1	Case 2	Case 3
Velocity (m/s)	1.4	0.9	1.1
Backpack weight (N)	220	63	315
Link1 (Spine to Hip) (cm)	9	8	9.6
Link2 (Hip to Knee) (cm)	44	43	45.4
Link3 (Knee to Ankle) (cm)	41.4	45	43
Link4 (Ankle to Football) (cm)	12	11.3	11.6
Joint1 (Hip)- lower limit (degrees)	-98.6	-111	-90
Joint1 (Hip)- upper limit (degrees)	2.2	6.5	4.5
Joint2 (Knee)- lower limit (degrees)	8	16	19
Joint2 (Knee)- upper limit (degrees)	146	127.2	112.3
Joint3 (Ankle)- lower limit (degrees)	12	7	16
Joint3 (Ankle)- upper limit	57	70	53

Since it is visually impossible to represent the training grid for all 12 input parameters, Figure 4.4 shows the two-dimensional plot for the 12 input parameters. The velocity and backpack weight parameters have the maximum effect on the outputs. Thus, these two parameters are shown in the x and y axes, respectively. In the figure, the small blue points represent the positions of the grid points (training points). The colored stars in the figure represent the studied testing cases. The green stars are for the on-grid cases; they are located exactly on these blue points. The off-grid cases are represented by the red stars. The figure visually shows how far each test case is from the grid points. However, this distance could be more or less in the real 12-dimensional grid space.

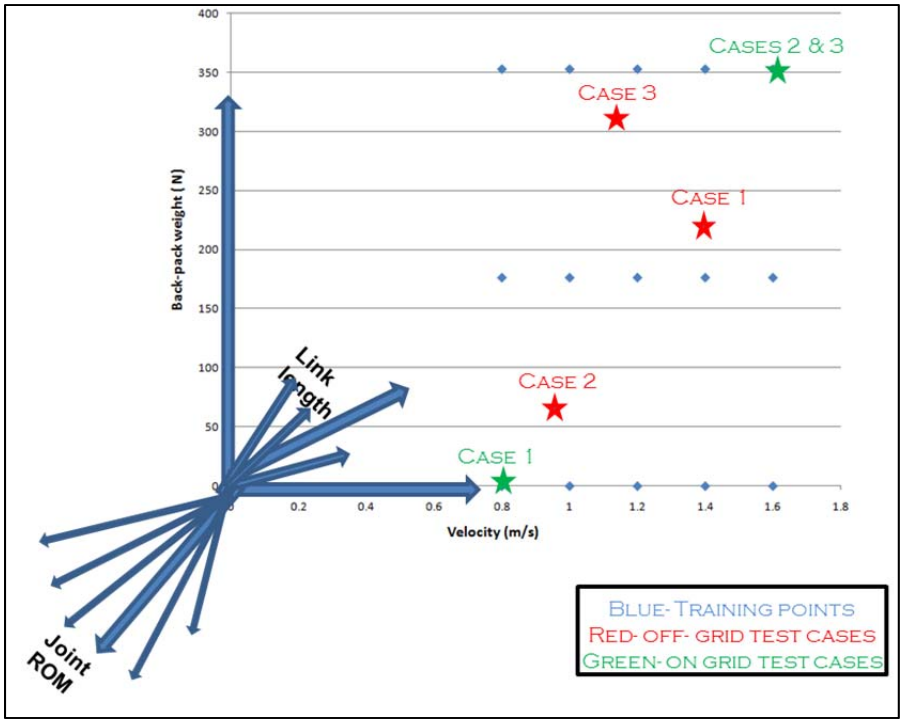

Figure 4.4: Two-dimensional plot for the training grid of the 12 input parameters in the walking task.

The comparisons for the testing cases were done in terms of resulting joint angle profiles, subjective (visual) motion, and joint torques. Each of those comparisons is included in the following separate subsections, in which they demonstrate both on- and off-grid testing cases. Plots of adjusted R-square values were drawn for the results (joint angles and torques) to demonstrate the accuracy of predicting the walking motion from the RGNN compared to PD.

Regarding the off-grid testing cases, the PD algorithm was run to obtain those cases to compare the results, but they were not included in the training process. From a speed point of view, the GRNN needed a fraction of a second to provide the output, while PD takes between 5 to 30 min, depending on the initial conditions in its optimization, to provide the solution.

4.3.1.1 Visual results

For any system, it is not enough to have accepted mathematical result without practical testing. Thus, let us represent the results visually to check whether the predicted motion results look realistic. On-grid testing cases are presented and compared first, followed by the off-grid cases. Figure 4.5 presents motion results for Case 1 of on-grid cases where both predicted GRNN and PD results are in the same figure. The exact PD testing case is shown in the upper part of the figure and the following figures, while the predicted motion from the GRNN is shown in the lower part of the figure. For each motion, snapshots were taken at 0, 33, 67, and 100% of the total task time.

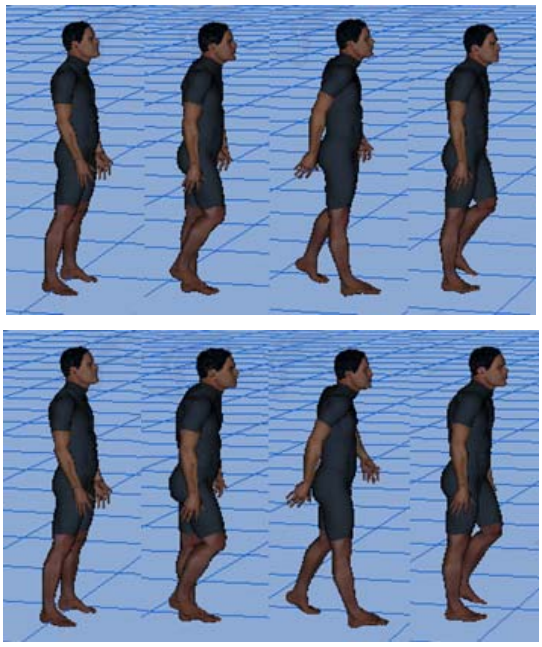

Figure 4.5: Visual results for on-grid Case 1 from PD and GRNN are shown in the upper and lower segments, respectively, for the walking task (0, 33, 67, and 100% of total task time).

In the second and third segments of the motion in the above case, Santos shows some differences in the predicted motion compared to the exact motion. There is some

bending in his back in the GRNN result, which is barely notable in this case. However, Santos's body parts have the same pauses over the task time. His legs, hands, and head are matched in the predicted and exact motions. This small error in predicting on-grid cases is because the network is trained to predict the task in general by having some trade-off between predicting both on- and off-grid cases in accepted results. To improve the prediction accuracy, the number of training cases should increase.

Case 2, which is shown in Figure 4.6, also shows matching between the predicted GRNN and exact PD motions. This case visually has more accurate results than Case 1. Velocity and backpack weight are at maximum values for this case, and that can be seen in the figure. Santos bends his back forward as a result of having 350 N on his back, and his step is larger than that in Case 1.

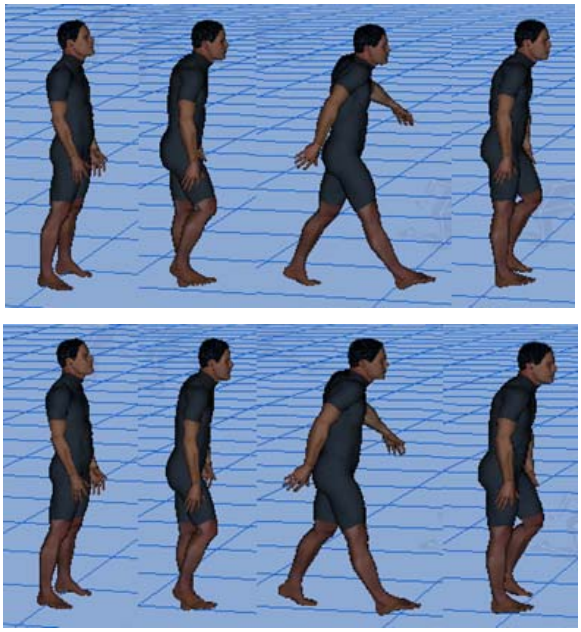

Figure 4.6: Visual results for on-grid Case 2 from PD and GRNN are shown in the upper and lower segments, respectively, for the walking task (0, 33, 67, and 100% of total task time).

From the two-dimensional training space plot in Figure 4.4, Case 3 has the same velocity and backpack weight values as Case 2. Figure 4.7 presents Case 3, which shows almost the same results as Case 2, because the only two notable factors, velocity and backpack weight, are exactly the same in both cases. These two cases differ in the last four ROM values, as shown in Table 4.5. On the other hand, there are slight differences between the two cases and the exact and predicted results for each case.

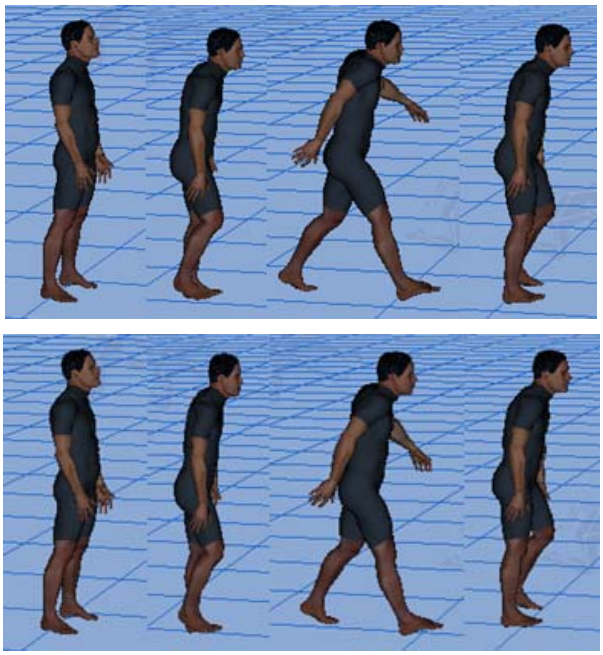

Figure 4.7: Visual results for on-grid Case 3 from PD and GRNN are shown in the upper and lower segments, respectively, for the walking task (0, 33, 67, and 100% of total task time).

The small differences in the predicted results from the GRNN and those from the PD are not critical. The GRNN provided accurate results for the on-grid cases; the cause and effect is notable in each case. The network also provided a solution in fractions of a second, which is an important improvement that could lead to on-line training for any task. Moreover, the overall motion produced by GRNN is visually acceptable.

Now, the three off-grid testing cases are compared visually and are expected to be less accurate than the on-grid cases, but should be accepted nevertheless. Case 1 is shown in Figure 4.8, which shows accurate predicted motion from the GRNN, which is very similar to the exact result. The visual result for this case is accurate and comparable to what was produced in the on-grid cases. Santos's body links also have an excellent match between GRNN and PD at the same time segment, as shown in the figure.

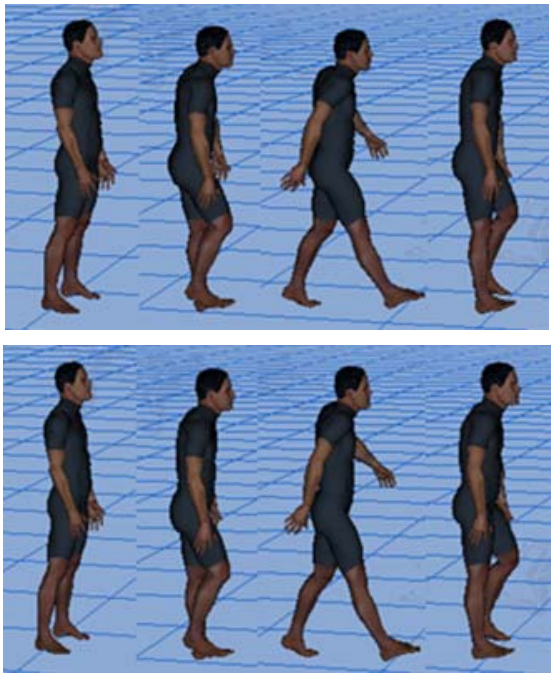

Figure 4.8: Visual results for off-grid Case 1 from PD and GRNN are shown in the upper and lower segments, respectively, for the walking task (0, 33, 67, and 100% of total task time).

Figure 4.9 shows the second case, Case 2, of off-grid testing cases. Santos's back is almost straight, which is a reflection of the case input where the backpack weight equals 63 N; the short step follows the velocity value, too. Only at the third segment of the motion is there some bending in Santos's back that is different from what it should be. This was a small error in prediction because it was not clear over the dynamic motion.

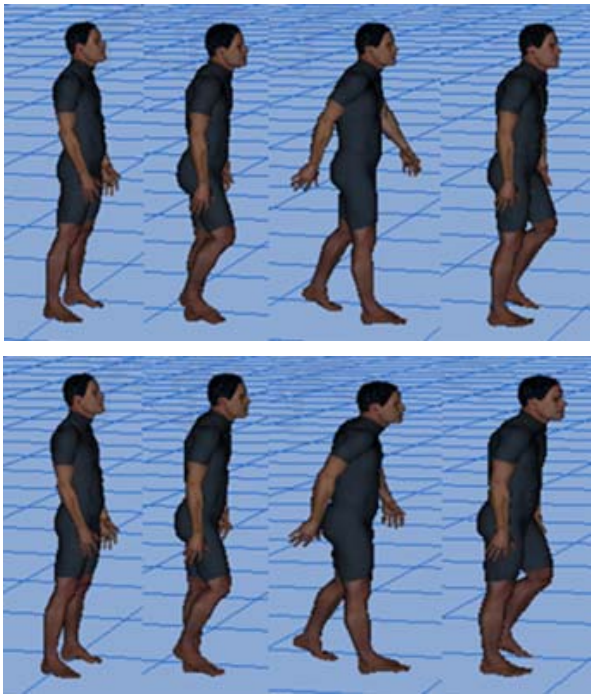

Figure 4.9: Visual results for off-grid Case 2 from PD and GRNN are shown in the upper and lower segments, respectively, for the walking task (0, 33, 67, and 100% of total task time).

Like the previous off-grid testing cases, the GRNN provided accepted and accurate results for Case 3, which is shown in Figure 4.10. Santos's back and hands might show some differences from the PD motion, but his motion is still correct and similar for both motions. Generally speaking, the visual results for predicting the walking with backpack task using GRNN was realistic and accurate (based on subjective validation). Santos was able to walk in a realistic manner in all shown testing cases. In this section, Santos's motions were compared, and his back is similarly bent for GRNN and PD in all cases. Even though there are two different types of outputs, the visual representation for the predicted joint control points was successful. To summarize, the real time and accurate prediction for any inputs combination is the main achievement of predicting this task. The network prediction ability should be similar for any on- and off-grid cases other than those presented in this study.

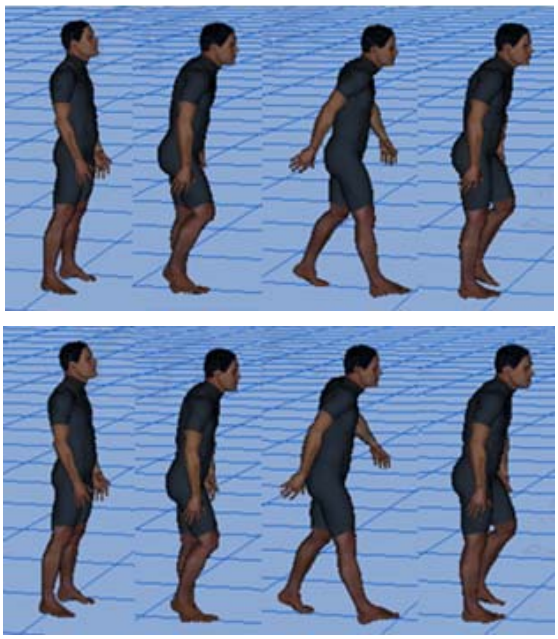

Figure 4.10: Visual results for off-grid Case 3 from PD and GRNN are shown in the upper and lower segments, respectively, for the walking task (0, 33, 67, and 100% of total task time).

4.3.1.2 Joint-angle profiles

From the total predicted outputs (i.e., the network outputs), which are 390 outputs, joint angle profiles (control point values) consist of 330 out of the 390 outputs. These 330 outputs represent the control point values for the 55 DOFs. These outputs are divided where each DOF has six values (control points) to shape the joint angle values over the motion time profile. In this part of the results section, an adjusted R-square value is plotted for those joint angle profiles between the predicted values from the GRNN and the exact ones from the PD algorithm. The plots statistically show the degree of the accuracy for the predicted outputs from the constructed network by presenting the R-square value for each case. The plots for the three on-grid testing cases are shown Figure 4.11, which also includes the adjusted R-square values for these cases. The plots are for the joint control point values (joint angle values) for the trained GRNN versus the exact values form the training cases.

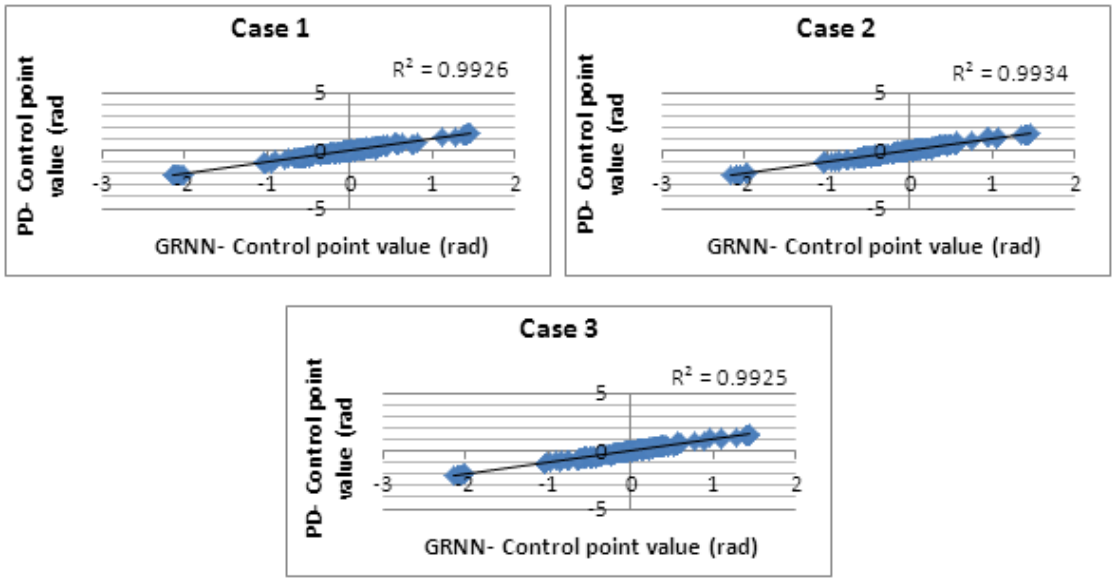

Figure 4.11: Adjusted R-square values for joint angle profiles at the three on-grid testing cases.

The adjusted R-square value was above 0.99 for all testing cases, as shown in the figure, which indicates that the network was able to predict joint profiles successfully with statistically high accepted results. A similar accuracy level is obtained for the other on-grid point. The presented cases are chosen, as mentioned, randomly from among the training cases, and the general prediction trend is similar for all training cases. On the other hand, these results were expected for the on-grid testing cases, because the network was trained to predict them (i.e., seen data).

Now, off-grid testing cases are compared in the same way. Figure 4.12 shows the adjusted R-square plots and values for the three off-grid testing cases. Surprisingly, similar highly accurate results were achieved for the off-grid testing cases, which had unseen, untrained input values. The adjusted R-square values also were above 0.99, suggesting that accurate correlations between different inputs were generated by the network.

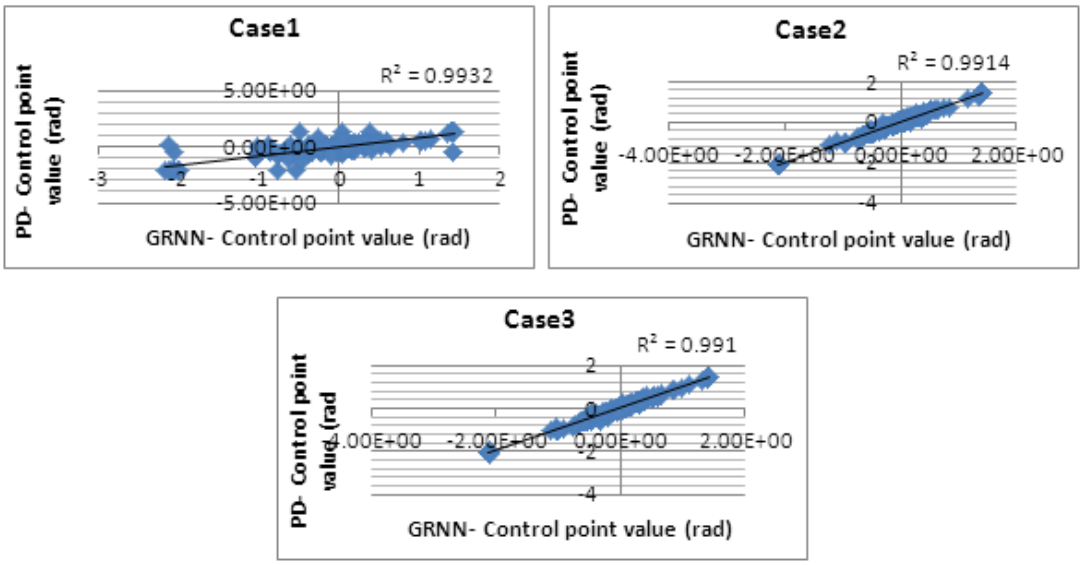

Figure 4.12: Adjusted R-square for joint control point values (55 DOF) of three off-grid testing cases.

In general, Figure 4.12 shows that the joint angle profile points on the accuracy lines were spread more than those in on-grid testing cases, especially for Case 1. Even though the R-square value for this case is similar to the others, this case shows worse network-predicted results because the points are distributed away from the accuracy line. Case 1 had that result because it had the furthest input values from the training cases (grid points) with respect to backpack weight and some joint angle limits, as shown in Figure 4.4 and Table 4.6. However, the visual result for Case 1, Figure 4.8, does not show any visual problem in matching the exact results.

From the previous results, it was found that the adjusted R-square values were high. The R-square values were around 0.99 for all cases even with that relatively large number of outputs. So, the network was generalized successfully, at this point, by providing outputs for the off-grid testing cases with high accuracy like the on-grid cases. Distribution of the joint angle profiles in the off-grid testing cases indicated some kind of inaccuracy between the predicted and exact results for the motion profile. However, the

visual results showed highly accepted results for these off-grid cases, which make the statistical results accepted too. In general, the trained GRNN showed a similar trend in predicting many other off-grid points.

4.3.1.3 Joint torques

This section discusses the second part of the results, which are some joint torques for this task. Those joint torques are the six torques at the lower-body joint DOFs (three for the hip, one for the knee, and two for the ankle), assuming symmetry, because they are the most highly articulated during the walking task. The joint torques are evaluated at ten time steps during the walking task, which resulted in 60 additional output values.

Plotting and calculating adjusted R-square values is the only way to compare the predicted results from the GRNN with those exact or actual values from the PD for this portion of the results. Starting with on-grid testing cases, Figure 4.13 shows those adjusted R-square values for the three on-grid testing cases in which each plot has all 60 output torque points.

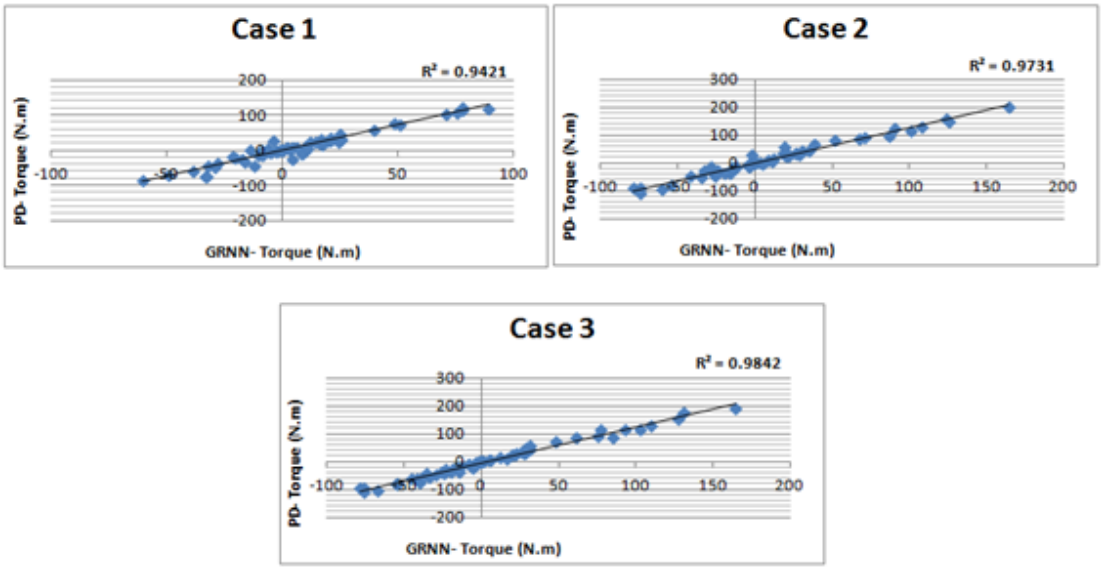

Figure 4.13: Adjusted R-square for six torque values with on-grid cases.

The points are mostly on or close to the accuracy line for all cases. The R-square values are all above 0.94, and the accuracy is acceptable in terms of the general output values. The predicted torques, however, were not quite as accurate as the joint angles, because each joint has a large range of torque values. Moreover, there are many ranges for torque values at different joints (i.e., some of them have a range of values between -500 to +500 N.m, while others are between -100 to +100 N.m). On the other hand, the joint angles are all measured in radian (rad), which is between +6.28 rad to -6.28 rad. For example, the torque values for Case 3 in Figure 4.13 are between around -80 to + 170, while the joint angle values for the same case in Figure 4.11 are around -1 to +2.

Hence, the GRNN could easily determine the range of the outputs for the joint angles and predict it with small error, but not with the torques. The general network prediction with small error leads to small error in joint angle prediction because the range of angle values is small and the produced error is small correspondingly. Large error is produced in torque prediction because their ranges are large and affected more than the angles by the prediction errors. In addition, the different ranges of torques that are at different joints increase the probability of error occurrence for the joint torque values. Regarding the accuracy among the torque plot results, Case 1 is found to have lower R-square value than the other two testing cases. The reason for that, again, is that Case 1 is further from the grid points than the other testing cases.

Figure 4.14 shows adjusted R-square values for the three off-grid testing cases between predicted torques from the network and actual PD output torques. The plots in the figure show that the points are close to the accuracy line in all cases, even though the accuracy was not very high and around 0.95 for Case 1 and Case 3 and 0.88 for Case 2. Joint angle profiles showed successful results visually even though they were more distributed than the torque values. Moreover, the adjusted R-square values are considered statistically high for those off-grid plots.

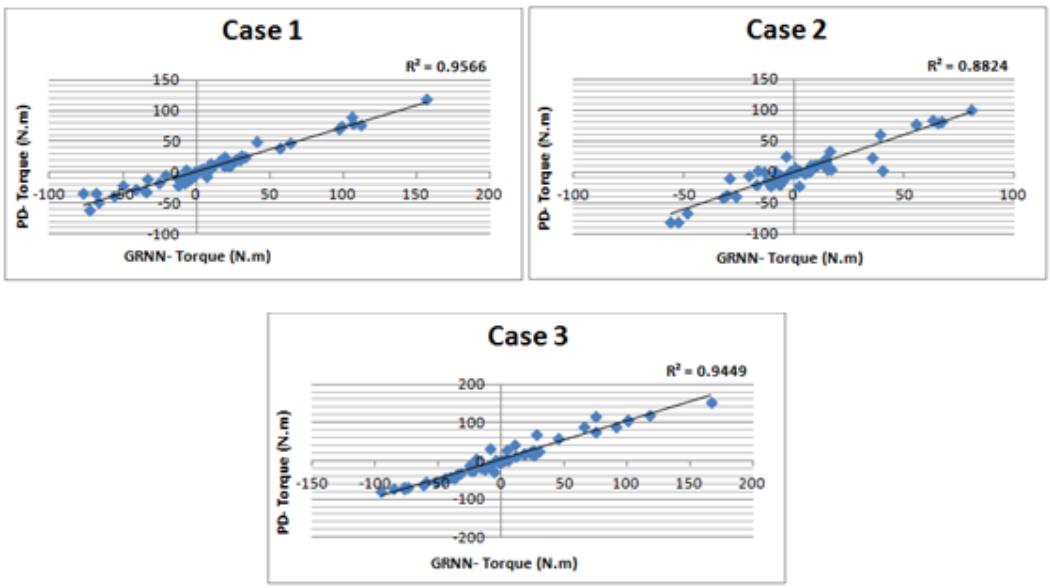

Figure 4.14: Adjusted R-square for six torque values (hip, knee, and ankle joints) for the three off-grid training cases.

Overall, the torque results are considered acceptable, especially because they are incorporated with other outputs in the same network. The point clusters or distributions were very close to the accuracy line, and the adjusted R-square values were acceptable. The general joint torque results have lower accuracy than the joint control points because of the variation in the joint torque values in the training cases. Having two different types of outputs also decreases the accuracy of the network prediction capabilities.

In summary, six different cases were studied, three each for on- and off-grid points, and the results of the GRNN were compared with those from the PD algorithm. The motion predicted using the GRNN was realistic and accurate (based on subjective validation). Since there are two types of outputs in the same network (joint angle profiles and joint torques), each was studied separately. Santos walked in an accepted way and showed similar outputs to the exact ones by subjectively comparing the joint angle results. Both comparisons showed success in predicting the motion for both tasks by the network. However, the accuracy of predicting joint torque control points was lower than

that in joint angle control points. In general, torque results were also accepted and comparable for both on- and off-grid testing cases.

4.3.2 Jumping up on a box

As described in the task formulation section, the task of jumping up on a box has fewer inputs than the walking task, which suggests that more accurate results are expected to be achieved. On the other hand, having two different types of outputs could decrease the accuracy of output prediction. In this section, four testing cases for the task of jumping up on a box are evaluated and analyzed: two on-grid cases and two off-grid cases. Like the walking task, the outputs of these testing cases are evaluated and compared between the GRNN predicted results and exact PD results.

On-grid cases are presented first to check the performance of the GRNN using the cases that it was trained to predict. The on-grid prediction should be accurate and match the PD results. These on-grid cases are shown in Table 4.7. These cases are chosen randomly from the training cases, because the network prediction ability for all on-grid cases is similar. So, testing any on-grid case is an indication of the general network prediction for all on-grid cases.

Table 4.7: Input variables for two on-grid testing points.

Input parameter	Case 1	Case 2
Box height (cm)	0.65	0.9
Link1 (Spine to Hip) (cm)	9	7.8
Link2 (Hip to Knee) (cm)	38	43
Link3 (Knee to Ankle) (cm)	39	39
Link4 (Ankle to Football) (cm)	9	12

Now, off-grid testing cases are presented for this task. These cases were expected to have very accurate results, but less accurate than the on-grid cases. Table 4.8 includes input values for both off-grid testing cases. These cases have input values that differ from the training values but are within the limits of the training space that were shown in the task definition section.

Table 4.8: Input variables for two off-grid testing points.

Input parameter	Case 1	Case 2
Box height (cm)	0.68	0.92
Link1 (Spine to Hip) (cm)	8.2	8.8
Link2 (Hip to Knee) (cm)	40	42
Link3 (Knee to Ankle) (cm)	39	39
Link4 (Ankle to Football) (cm)	10	11

Like in the walking task, these off-grid cases are selected randomly, where there is an infinite number of testing cases. These two cases are chosen to have input values within the grid space but furthest from the training cases. In other words, the box height in the training cases have a 5 cm increment between the training cases (see Table B.2). Thus, those testing cases with box heights that equal 0.92 and 0.68 are further from the grid points than 0.61 or 0.99. The other inputs also have values in between the training values, but not the training values themselves. Therefore, these chosen cases are relatively some of the furthest points from the training cases, and should have the least accurate predicted outputs from the network. So, predicting these cases well means that the all other possible inputs are also predicted well, even better than the results of these cases.

4.3.2.1 Visual results

Each one of the presented on- and off-grid testing cases is visually compared by comparing the results between the predicted motion profile from the GRNN and the exact motion from the PD algorithm. As shown in Table 4.7, there are two on-grid testing cases. Case 2 has a box height larger than Case 1, and this will be evident in the visual results for both cases. Figure 4.15 presents the visual result for Case 1. The upper and lower parts represent motion segments over the task time for GRNN and PD, respectively. The total task time is approximately 1 second, and segments are taken at 0, 25, 50, 75, and 100% of total time. Even though there are some differences in the scale of the snapshots for both motions, they actually have the same behavior over the task time, including the height of the feet and hand positions. The behavior was evaluated visually, and Santos performed the task in the exact way in both results.

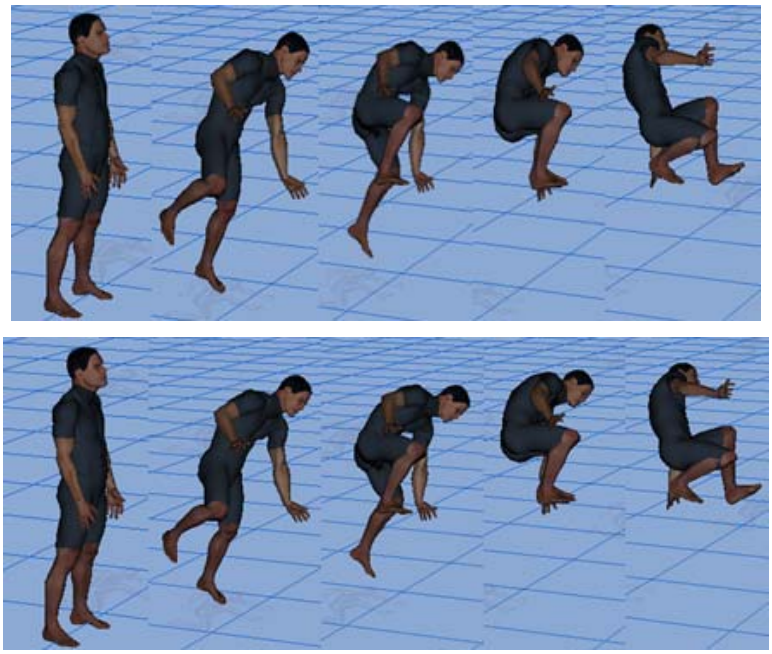

Figure 4.15: Visual results for Case 1 on-grid from GRNN and PD over the motion profile of the task (0, 25, 50, 75, and 100% of total task time). GRNN is shown in the upper portion of the figure.

Similar results are noticed in Case 2, which is shown in Figure 4.16. The difference in the reached height at the end of the task between Case 1 and 2 is evident; it is higher in Case 1. There were no differences in the motions from GRNN and PD in Case 2.

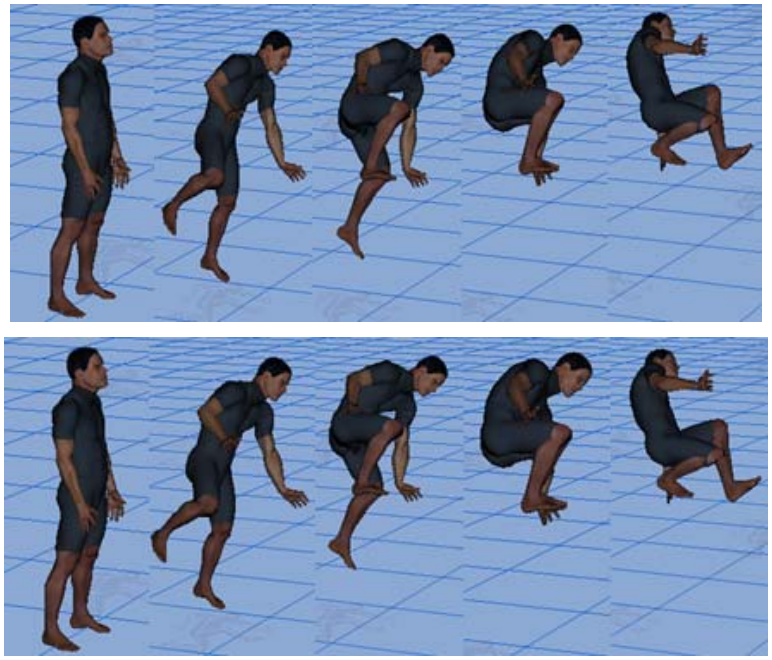

Figure 4.16: Visual results for Case 2 on-grid from GRNN and PD over the motion profile of the task (0, 25, 50, 75, and 100% of total task time). GRNN is shown in the upper portion of the figure.

As shown in Table 4.8, there are two off-grid testing cases. The off-grid cases are compared in the same way the on-grid cases were tested. These two off-grid testing cases are shown in Figures 4.17 and 4.18, where both figures indicate identical results from both GRNN and PD. In Case 1, which is shown in Figure 4.17, there is complete matching between GRNN and PD over all motion segments. All body motions were identical in terms of Santos's limbs and back as well as his final position.

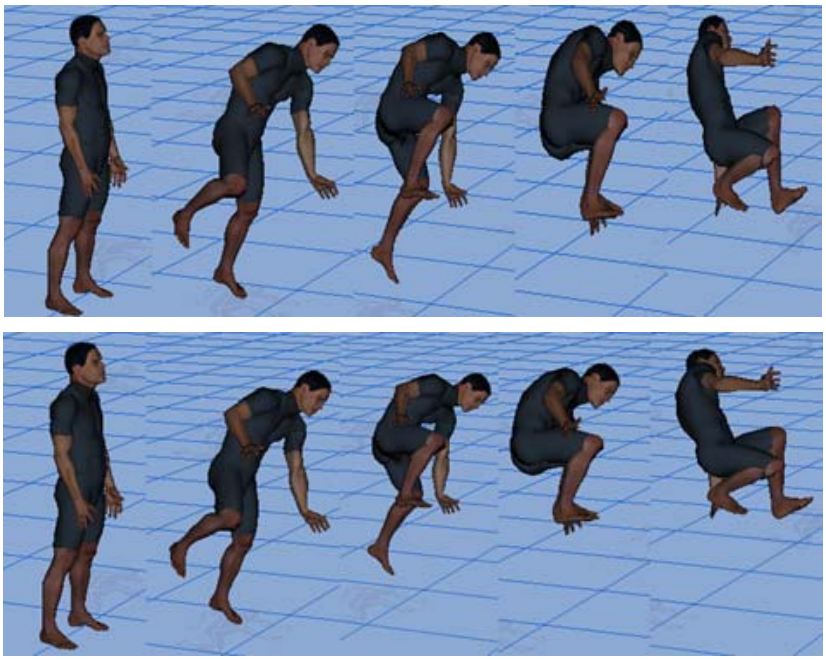

Figure 4.17: Visual results for Case 1 off-grid from GRNN and PD over the motion profile of the task (0, 25, 50, 75, and 100% of total task time). GRNN is shown in the upper portion of the figure.

Figure 4.18 shows the GRNN and PD motion results for Case 2. Santos's hands and feet from the predicted motion were at the exact locations that the PD results provide. Throughout the motion times that are shown in Figure 4.18, his feet, hands, and all other body segments match the PD motion result. In general, prediction of all on- and off-grid cases was visually successful and accurate as far as the final positions for Santos's hands and feet. The network was able to predict the motion for all testing cases very accurately, which means that this ability is also applied for any other testing case. This enhances the GRNN's ability to predict tasks with very accurate performance requirements like this task. There are contact points (for the hands with box, and the feet with ground and box) that Santos should touch exactly and accurately, and the network successfully achieved that.

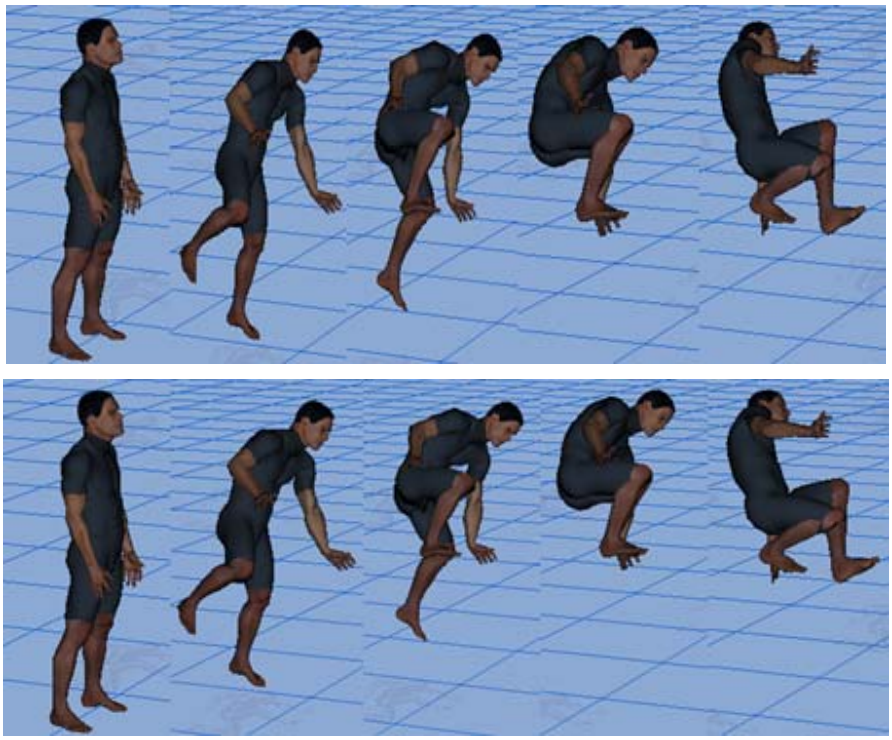

Figure 4.18: Visual results for Case 2 off-grid from GRNN and PD over the motion profile of the task (0, 25, 50, 75, and 100% of total task time). GRNN is shown in the upper portion of the figure.

4.3.2.2 Joint-angle profiles

The visual results showed a high matching between the predicted motion from the GRNN and the exact PD motion. Like in the walking task, this section compares the accuracy of the network prediction for the task of jumping up on a box statistically. Figure 4.19 shows the on-grid testing cases that were presented above in the visual results section. Those cases should have more accurate results than the off-grid ones, because the network was trained to predict them exactly and adjusted correspondingly. The adjusted R-square values for these cases are high, as shown in the figure. The predicted points are all on the accuracy line or close to it at both cases.

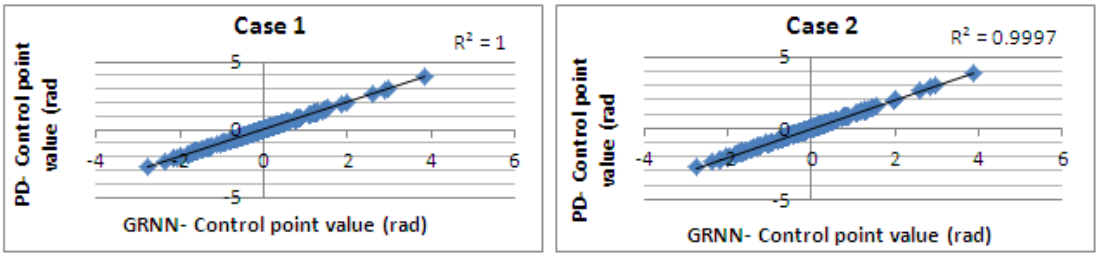

Figure 4.19: Adjusted R-square for the joint control point values of the two on-grid testing cases.

Figure 4.20 presents adjusted R-square values for the two off-grid testing cases. Both cases have adjusted R-square value around 1, which show the high degree of accuracy in predicting the both cases. The high R-square values for off-grid testing cases because the task has small number of inputs (5-inputs) which allows the network to predict the task accurately. The plot results match with what was seen in the visual figures above, which also show subjective matching between the predicted and exact results. For each plot, the axes represent GRNN joint splines (joint control points) at the horizontal line and PD joint control points for the vertical line.

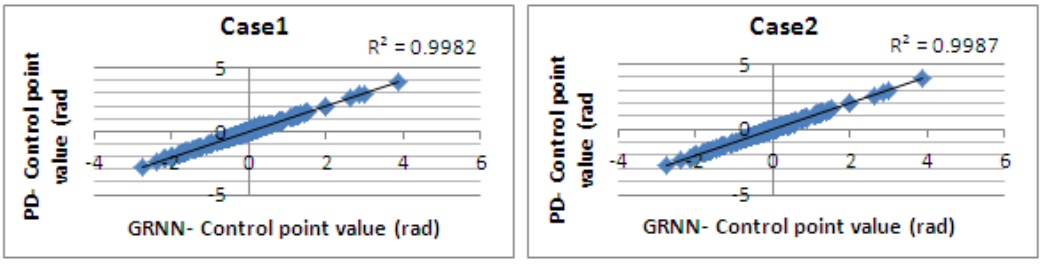

Figure 4.20: Adjusted R-square for the joint control point values of the two off-grid testing cases.

In general, adjusted R-square values for joint control points are relatively high in all presented motions in this task. All on- and off-grid testing cases were predicted

accurately, where all adjusted R-square values were either 1 or around that. These results are obtained in more accuracy than for the walking task, because the jumping on a box task has a smaller number of inputs, and small increments for the inputs between the different collected training cases.

4.3.2.3 Ground reaction force (GRF)

The second part of the network's predicted outputs is the maximum GRFs on both feet. As described in the task definition in section 4.2.2, the network was trained to predict the peak values of the GRFs at each foot. The peak values are presented by 20 points or a sample of times that have the maximum GRFs on the foot over the task time. So, there are 40 outputs from the network that represent the 20 maximum GRFs at the right and left feet, respectively. To show how the GRFs are presented in this task, Figure 4.21 shows the predicted GRFs from the GRNN on the left and right feet for Case 1 in off-grid testing cases. The vertical axis represents the force values in (N), while the horizontal one is for a sample number. The predicted samples are always arranged so that sample number 10 has the maximum value (the peak GRF value). The plots have different scales because the GRF on both feet are sharply different.

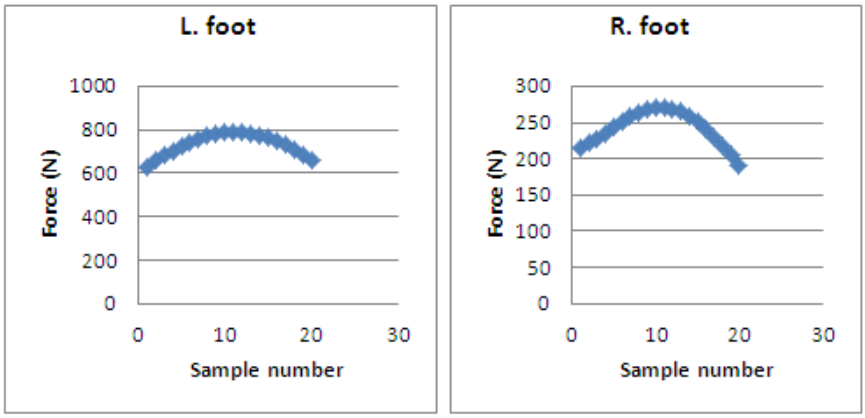

Figure 4.21: The maximum 20 GRF values over the task time for the predicted off-grid testing Case 1.

Like joint angle control points comparison, the accuracy of the predicted GRF values from the GRNN should be checked and evaluated. This section presents the adjusted R-square values for the predicted GRFs versus exact PD results for all on- and off-grid testing cases. The on-grid testing cases are evaluated first in Figure 4.22. Both adjusted R-square values are equal to 1, which means there is a complete match between the exact PD and predicted GRNN outputs for GRFs.

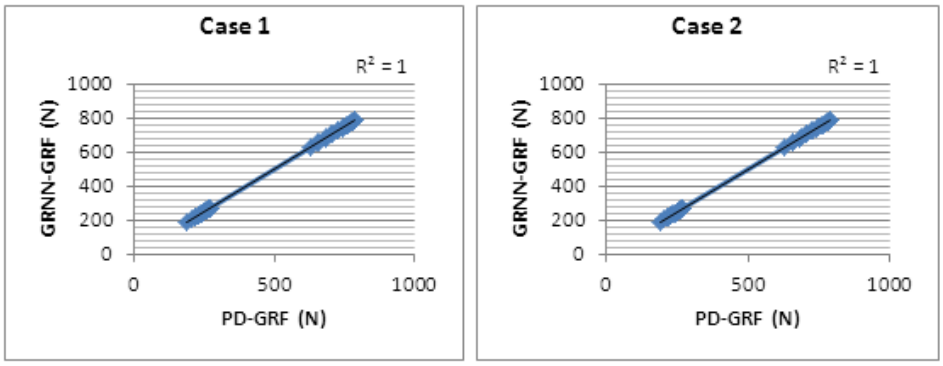

Figure 4.22: Adjusted R-square for GRF values (at right and left feet) between PD and GRNN for two on-grid testing cases.

The adjusted R-square values in the figure present outstanding prediction for GRFs and superior accurate results from those achieved for joint torques in the walking task. Both on-grid cases have completely accurate results, R-square equals 1 for both cases. This high accuracy result indicates that even the off-grid results are accurate too. The outstanding accuracy for the on-grid testing cases are obtained because of the following reasons:

1. Using a smaller number of inputs, which produces lower dimension for the task (less grid space). In lower grid dimension, the training points on the grid are closer to each other, which increases the accuracy of prediction.

2. Using small increments between different training cases. Thus, even the prediction for the points in between the training cases becomes more accurate.
3. Resulting GRFs from different training cases are similar (have low variation). So, the low variation allows the network to be trained to predict these outputs better.

After testing the on-grid cases, off-grid cases are presented in Figure 4.23. R-square values for these off-grid testing cases are high. The R-square value in Case 1 is around 1, while it is exactly 1 in Case 2. The value of 1 for R-square in Cases 2 occurs because the GRF values have small variations over the training grid, and so the network predicts them accurately. In the figures of on- and off-grid testing cases, there are two separate clusters located on the accuracy line. Those clusters represent the left and right feet GRFs, where the ranges mentioned above are noticed and successfully handled by the network.

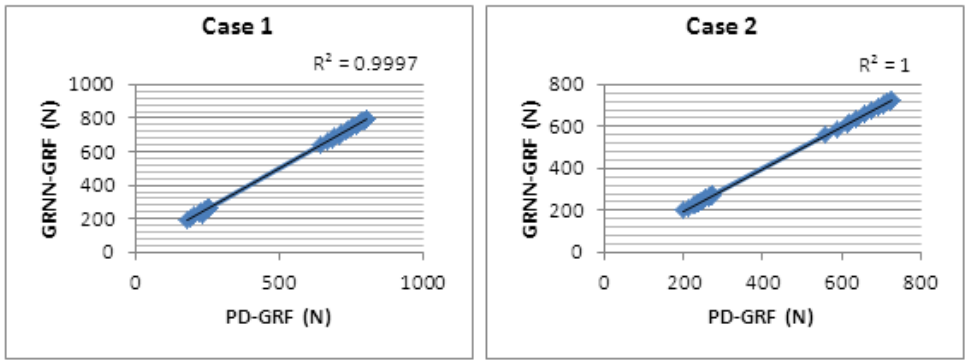

Figure 4.23: Adjusted R-square for GRF values (at right and left feet) between PD and GRNN for two off-grid testing cases.

Unlike the accuracy achieved for joint torque control points in the walking task, adjusted R-square values for the GRFs in this task show highly accurate results. The reasons for this accuracy were already mentioned above. As seen in the GRFs plots, all predicted points were exactly on the accuracy line for all cases.

In summary, the obtained accuracy for predicting this task was higher than that in the walking task. For this task, four different cases were studied, two each for on- and off-grid points, and the results of the GRNN were compared with those from the PD. Having two different types of output did not affect the accuracy of predicting both types. Unlike in the walking task, predicting both outputs was highly accurate in this task. This task already required having higher accuracy to solve the contact point issue, where the network did not have any problems in predicting those contact points (locations of the contact between the hand and the box and the feet and the box). In conclusion, this task was an indication of the outstanding accuracy that can be achieved by the GRNN if it is trained well for such a task (having small increments between input values in the training grid).

4.4. Discussion

In this chapter, the new presented methodologies in Chapter 3 were applied to predict human motion prediction using GRNN. Constructing, training, and selecting the GW, as well as testing the GRNN network were all performed automatically to maximize the prediction ability for two motion tasks: walking with a backpack and jumping up on a box. Each one of these tasks had its own GRNN that was automatically constructed with maximum performance for that task. The predicted results from the constructed GRNNs were tested and compared with the exact results from PD to evaluate the network performance. Predicting the results of both tasks was subjectively and objectively accurate and fast. The accurate results were also related to the deterministic nature of predictive dynamics.

The studied motion tasks were chosen to examine GRNN's ability to address different issues. The walking task has a relatively large number of outputs and 12 inputs, and there are two different types of outputs. The network was able to accurately predict all outputs from the different types of this task. Jumping up on a box is a complicated

task with many constraints. This task also needs high accuracy in predicting its outputs, because there are contact points between the hands and feet with the box over the task time. The constructed network for this task addressed all these issues successfully with outstanding fast and accurate results.

The general trend in predicting the presented motion tasks using GRNN was producing fast and accurate results for any new input combination within the grid space (training space). This trend applied when any on- or off-grid input was fed into the network. The results from this chapter prove the assumption of using ANN to predict human motions realistically and quickly. The issues in the presented tasks in this chapter were addressed successfully by the GRNN. Thus, the use of GRNN might be generalized over many other motion tasks. Therefore, there is a great potential for GRNN to be widely used in real-time, task-based motion prediction and to perform like a human brain, developing and training continuously.

Applying the new methodology for selecting the network GW depending on the task was valid and logical. It was validated by the successful testing of various on- and off-grid results. The selected GW value at each task also reflected the dimension of that task (number of inputs), where larger input space needs larger GW value to cover the grid space properly. The best GW value equaled 0.45 for walking task, while it was 0.05 for jumping up on a box.

The two predicted tasks in this chapter had different types of outputs. The results of predicting these different outputs showed that having two different types of outputs decreases the general prediction capabilities for the network. However, the accuracy of output prediction depends on the variation in the values of that output. The predicted torque values in the walking task had the least accurate results because their values had the largest variation in the training cases.

In the jumping up on a box task, the high accuracy in predicting the task results points out the usefulness of GRNN for DHM applications. This task is more complicated

than the walking task, but the network predicts it more accurately because this task has a smaller number of inputs. Therefore, training the network to accurately predict such a complex task is doable once the network is trained well and the task inputs are well defined.

On the other hand, there are some challenges and limitations for using ANN in motion prediction, including: 1) professional and automatic task definition, 2) the optimal number of training cases for a task, and 3) extrapolation of prediction capability for off-grid points. The third limitation, however, is not critical at this point because we used extreme combinations of inputs during the training stage so the network could produce acceptable results for any feasible inputs.

CHAPTER V

NEURAL NETWORK-BASED POSTURE PREDICTION

5.1. Introduction

Researchers have studied many aspects related to human posture prediction in which many factors take part and control production of realistic posture. Since people simply behave differently, some strategies were found to study how and why people choose their postures in order to reproduce these postures correctly. However, human postures are considered learned skills, where there are an infinite number of postures for an infinite number of conditions and tasks. Hence, studying posture prediction in the general scope is critical to eventually understanding the strategies that people use when they intend to do a task.

Human posture prediction was developed based on two main approaches. The first one involves prerecorded data using motion capture systems combined with anthropometric data and functional regression models. The second approach involves real-time inverse kinematic optimization-based posture prediction based on some objective functions (performance measures, or PMs). The conceptual formulation of the second approach is presented in the following optimization problem:

Find: Joint angles (5.1)

To minimize: One or more performance measure(s) (e.g., joint torque, joint displacement, etc.).

Subject to: Distance between finger-tip (end effector) and target point=very small value.

On the other hand, showing cause and effect in the motion capture approach is limited. For example, if some joint range of motion (ROM) changes, the produced posture does not change. In the inverse-kinematic approach, more cause and effect is obtained. However, this approach does not guarantee realistic posture results for all tasks and conditions because it depends on solving an optimization problem that could produce

poor results for some problems. Therefore, both approaches might be used as bases for some prediction tools to predict more generalized and solid human postures. As one of these potential tools, artificial neural network (ANN) has powerful prediction capability. This capability could be combined with either one of the developed approaches to develop a more mature posture prediction strategy that works well for any task under any condition.

Generally, predicting human postures based on the task to be accomplished would lead to understanding and extracting the factors or controls that drive human performance (performance measures, PMs). This extraction might be achieved using the network that predicts postures by either connecting the network neuron values to these performance measures, or predicting these performance measures directly as outputs from the network. The previous work for ANN in posture prediction showed promise. Studies used different types of ANN, but not the GRNN. Thus, it is important to evaluate GRNN performance in the context of optimization-based posture prediction problems. So, this thesis presents the use of GRNN to predict human posture for the following reasons:

1. To further investigate potential issues when using GRNN to simulate tasks that involve contact constraints or other conditions involving Cartesian locations.
2. To provide initial work in posture prediction to discover the limitations on using GRNN in this digital human modeling (DHM) problem.
3. To provide an initial investigation providing a platform for further study of ANN with posture prediction (i.e., evaluation of performance-measure combinations).
4. To demonstrate the feasibility of using a large number of outputs (including the use of joint angles as well as joint torques).

In this chapter, GRNN is used to predict human posture for two tasks: touching a point in front of the body with and without external force on the hand. This study also provides initial and promising work toward having a posture prediction strategy that guarantees provision of real postures for any task under any condition. The network

outputs are the joint angles in the task of touching a point in front of the body without external force. The joint angles and torques are the network outputs in the task of touching a point with external force. This study is superior to previous studies on posture prediction using ANN because it has a large number of outputs for the upper body, increasing the accuracy of mimicking the real human body in Santos's design. The following contributions are achieved in this chapter:

1. Two task-based posture predictions, touching point with and without external load, for a 55-DOF human model using ANN.
2. Incorporation of joint torques as part of the predicted outputs.
3. The first use of GRNN in task-based posture prediction with highly reasonable results.
4. Prediction of a relatively large number of outputs with different types (joint angles and torques) for a 55-DOF human model.

5.2. Proposed Task Definition, Training Process, and Network Properties

This section presents the definition for the tasks that will be used to train the GRNN for posture prediction. In addition, the training process and the network's inputs, outputs, and properties will be described. There are separate subsections for both applied tasks, posture prediction for touching a point in front of the body with and without external force.

Collecting training data for both tasks was done using the Santos software. The software has a well-developed and fast posture prediction tool that is used to predict points around the body and predict them with extra conditions like external loads. Posture prediction in the Santos software is achieved by solving the optimization problem that was shown in the introduction section (Equation 5.1). The main changes in this problem between different tasks include: 1) the performance measure(s) that are used to be minimized and 2) the number and types of the constraints that are added to or removed

from the problem. The work of collecting the training cases and training the network was done on a Windows 7 computer with an Intel ® Core™ 2 processor and 8 GB of RAM.

5.2.1 Posture prediction for touching a point without external load

This task is defined as touching a point in front of the body with the right hand with some conditions. These conditions include the use of joint displacement PM as cost function in an optimization problem (Equation 5.1), and freezing the hip and lower body in posture prediction, which is chosen to simplify the problem. Thus, the upper part of the body is the only part that changes when predicting any posture. Target points were selected in order to test the feasibility of using ANN for posture prediction, rather than testing the accuracy of posture prediction itself. Thus, targets in front of the avatar were used, because in general, they are easier to reach and yield more realistic postures. In this way, we isolated the application of ANN as the point of study. The joint displacement is chosen to be minimized in Equation 5.1, because it is a fundamental objective function that is used generally in the literature for the task of reaching targets on the front side of the body (Yang et al., 2004).

The network used 31 training cases, which were randomly collected to cover all reachable points in front of the body, as shown in Figure 5.1. The furthest points from Santos's body represent the maximum distance that his hand could reach without moving his hip. The training points are shown in small red balls in the figure. There are 9 input parameters for this task, including: 1) point or target position in three-dimensional space (X, Y, and Z) and 2) ROMs for three DOFs. Those ROMs were shoulder flexion-extension, shoulder abduction-adduction, and elbow flexion-extension. Those DOFs were included because they are the most significant DOFs for reaching the most target points.

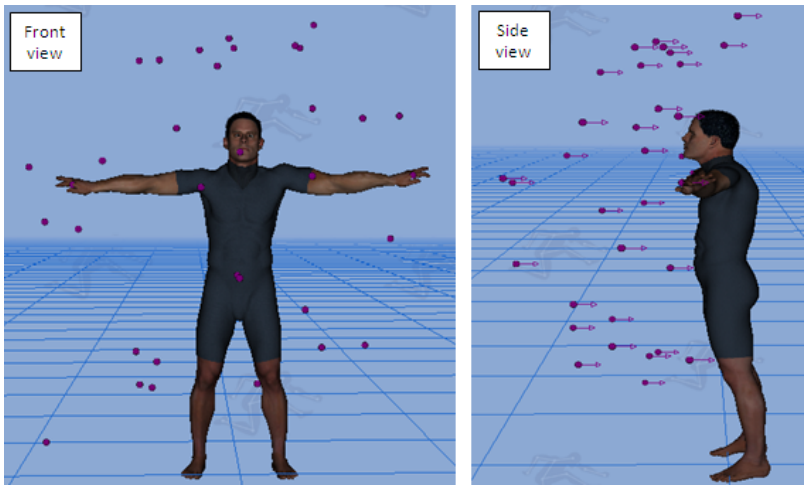

Figure 5.1: Santos with the points that are used in the training cases, shown in red.

Table 5.1 includes all input parameters for the network with their values that were used to train the network. Lower and upper terms in the table indicate the lower and upper ROM limits for each DOF. R.Shoulder1 and R.Shoulder2 represent shoulder flexion-extension and shoulder abduction-adduction, respectively. Value 1 in the ROMs represents the default ROM's values for a typical person, while Value 2 in the ROMs is chosen to be smaller than Value 1 to reduce the upper and lower ROM for each DOF. The input combinations for all training cases in this task are shown in Table B.3 (Appendix B). Value 1 and Value 2 have different meanings between target position and ROMs. For target position (X, Y, and Z), Value 1 represents the minimum value of the three-dimensional target position that was used in creating the training cases, and Value 2 represents the maximum value of the three-dimensional target position that was used in creating the training cases. So, the value of the target position (X, Y, and Z) in the produced training cases is between the range of the two presented minimum and maximum values (Value 1 and Value 2). Regarding the ROMs, some training cases were produced using Value 1 of ROMs and others with Value 2. There was no training case created using mixed values between Value 1 and Value 2 for any ROM. So, all ROMs values in each training case were either Value 1 or Value 2.

Table 5.1: The network's training values for the input parameters in the task of touching a point without external force.

Input parameter		Value 1	Value 2
Target position (cm):	X	-102	77
	Y	-56	120
	Z	-98	3
R. Shoulder1- lower (degrees)		-23	-5
R. Shoulder1- upper (degrees)		123.5	60
R. Shoulder2- lower (degrees)		-19	-5
R. Shoulder2- upper (degrees)		111	50
R. Elbow- lower (degrees)		-148.5	-80
R. Elbow- upper (degrees)		-12.5	-40

Outputs of this task include upper-body joint angles, which are 41 DOFs, because the lower-body DOFs were frozen in the task definition. Thus, the DOFs for the lower body are not changed at any produced posture. The GW in the created network is determined using the method described in Chapter 3 and equals 0.25. The resulting GW value is relatively large for a task requiring highly accurate prediction ability from the network, touching a point exactly in the space. However, this value is reasonable for this task, because there were only 31 points collected for training from the whole reachable zone in the front of the body. Thus, there are many gaps (empty spaces) between the training cases. Consequently, the GW must be large enough to successfully predict the points that are located in these gaps. However, a large GW can decrease the accuracy of predicting on- and off-grid points in general.

In summary, the constructed network for this task has 9 inputs and 41 outputs and 31 training cases. The GW value that provides the best network performance equals 0.25. The summary of this task and constructed network properties are shown in Table 5.2.

Table 5.2: The touching point without external force task definition and constructed network properties.

Task Definition	- Touching point in front of the body without external force on hand - Task variables: • Target point position (x, y, z) • Joint ROMs (upper and lower limits) ➢ R. Shoulder 1: Lower (-23, -5); Upper (123.5, 60) ➢ R. Shoulder 2: Lower (-19, -5); Upper (111, 50) ➢ R. Elbow 1: Lower (-148.5, -80); Upper (-12.5, -40)
Network properties	- Inputs: 9 (Target position (x, y, z) & ROMs) - Outputs: 41 (Upper body joints) - Training cases: 31 case - Gaussian width= 0.25

5.2.2 Posture prediction for touching point with external load

The task of touching a point on the front side of the body with external force differs from the task that was defined above. This task has some other conditions that are applied on the task definitions and more outputs. There is an external load of 100 N applied on Santos's right hand, which is the hand reaching toward the target point. Two performance measures are used to optimize Santos's posture prediction problem in Equation 5.1: joint displacement and maximum joint torques in weights of 25% and 75%, respectively (Marler et al., 2011).

Like the first task (touching a point without external force), the input parameters for this task include: 1) point or target position in three-dimensional space (X, Y, and Z) and 2) ROMs for three DOFs. The input values in the training cases, however, were different from those in the first task. Other conditions on this task are: 1) using two performance measures (joint displacement and maximum torques), 2) active balance constraint (zero moment point), and 3) frozen hip and lower body. The zero moment point condition is responsible for providing only postures with balance; any imbalanced postures are infeasible solutions (Marler et al., 2011). Training cases were collected that they cover all reachable points on the front side of the body, which was similar to the

space shown in Figure 5.1. There were 34 training cases, where Table 5.3 shows the input parameters and their training values for this task. The input combinations for all training cases in this task are shown in Table B.4 (Appendix B).

Table 5.3: The network's training values for the input parameters in the task of touching a point with external force.

Input parameter		Value 1	Value 2
Target position (cm):	X	-76	59
	Y	-47	107
	Z	-82	-13
R. Shoulder1- lower (degrees)		-23	-5
R. Shoulder1- upper (degrees)		123.5	50
R. Shoulder2- lower (degrees)		-19	0
R. Shoulder2- upper (degrees)		111	50
R. Elbow- lower (degrees)		-148.5	-70
R. Elbow- upper (degrees)		-12.5	-30

As mentioned, the values for the input parameters are different from those in the first task to have more variation in the training process in terms of changing joint ROMs as well as target point locations. On the other hand, the Value 1 in all ROMs are the same for both tasks because those are the default ROM values for a typical person. Value 2 in the ROMs is chosen to be smaller than Value 1 to reduce the upper and lower ROM for each DOF more than that in the first task.

The task outputs include upper-body joint angles, which are 41 DOFs, and 47 represent all body joint torques excluding the eyes' DOFs. As described above, the joint angle outputs include the upper portion of the body, since the lower part and the hip were

fixed when the task was specified. As indicated, this task has more outputs than the first task with a different range of values, which adds extra competition for the network prediction accuracy. So, the task has 88 outputs in total: 1) 41 upper-body DOFs (same previous task outputs) and 2) 47 body joint torques. It was not surprising to predict all body joint torques in this task because all joint torques were affected and changed by having external forces at any posture even if those joints are not moving when posture is predicted. For example, when somebody carries a load on his hand, the ankle joint angle does not change, but the torque changes.

Thirty-four training cases were used to train the network in this task. It had best network prediction when GW equaled 0.3, which was, again, found automatically during the training process. The GW value was found to be close to that in the first task, which was touching a point without an external load. Both tasks have slightly different GW values because of having a different number of training cases and outputs. Using more training cases produces fewer gaps in the training grid. Predicting more outputs, however, decreases the accuracy of the predicted outputs, especially when there are two different types of output. The summary of this task and constructed network properties are shown in Table 5.4.

Table 5.4: Touching a point without external force task definition and constructed network properties.

Task Definition	- Touching point in front of the body with external force on hand - Task variables: • Target point position (x, y, z) • Joint ROMs (upper and lower limits) ➢ R. Shoulder 1: Lower (-23, -5); Upper (123.5, 50) ➢ R. Shoulder 2: Lower (-19, 0); Upper (111, 50) ➢ R. Elbow 1: Lower (-148.5, -70); Upper (-12.5, -30)
Network properties	- Inputs: 9 (Target position (x, y, z) & ROMs) - Outputs: 88 (Upper body joints) ➢ Upper body joints= 41 ➢ All body joint torques= 47 - Training cases= 34 - Gaussian width= 0.3

5.3. Results

This section illustrates and discusses the postures that resulted from the GRNN prediction. In addition, these results are compared with the results from posture prediction. Each application will be presented separately and evaluated statistically and visually. In the first application, touching a point without external force, there are three testing cases for each on- and off-grid point. The second application, touching a point with external load, has two testing cases for each on- and off-grid case, because we found from applications in Chapter 4 that the general trend in the network prediction ability for all on- and off-grid cases is similar. Thus, it is enough to present only two cases for each on- and off-grid case to measure the degree of success in the task prediction in general.

5.3.1 Posture prediction for touching point without external load

After the training process was completed, a GRNN could be created for this specific task to predict any new input values within the grid space (inside the training space). The network prediction ability was tested for both on-grid and off-grid points (each point represents a combination of input parameters that were described in Section 5.2.1). In this section, three on-grid points as well as off-grid points are tested, and the network outputs are compared with those from Santos posture prediction (the actual outputs that were used in the training process).

Each on- and off-grid testing case is presented in separate tables. Table 5.5 includes the input parameters for the three on-grid testing points. These cases are chosen randomly, because the accuracy of predicting some on-grid points is similar for all on-grid points. As shown in the table, all input values are the same values that were used to train the network. Those cases should be predicted accurately by the network. All input parameters were already described above.

Table 5.5: Input parameter values for the three on-grid testing cases in the task of touching a point without external force.

Input parameter		Case 1	Case 2	Case 3
Target position (cm):	X	-1	77	-102
	Y	56	74	49
	Z	-37	-21	-22
R. Shoulder1- lower (degrees)		-19	-19	-19
R. Shoulder1- upper (degrees)		111	111	111
R. Shoulder2- lower (degrees)		-5	-23	-5
R. Shoulder2- upper (degrees)		60	123.5	60
R. Elbow- lower (degrees)		-148.5	-80	-148.5
R. Elbow- upper (degrees)		-12.5	-40	-12.5

Regarding off-grid testing cases, the network should be able to predict them because they belong to the same range of values that it was trained on. Those off-grid points are selected randomly and are shown in Table 5.6. Most of the input values for those cases are not the same values that were used in the training. These cases, however, are still within the range of values that were used in the training (inside the training grid).

The results for all on- and off-grid testing cases are presented in the following sub-sections, which are divided based on the type of results (visual or statistical) compared.

Table 5.6: Input parameters for the three off-grid testing cases in the task of touching a point without external force.

Input parameter		Case 1	Case 2	Case 3
Target position (cm):	X	-8	-62	42
	Y	75	24	76
	Z	-79	-69	-44
R. Shoulder1- lower (degrees)		-19	-6	-10
R. Shoulder1- upper (degrees)		111	100	80
R. Shoulder2- lower (degrees)		-23	-21	-18
R. Shoulder2- upper (degrees)		123.3	86	110
R. Elbow- lower (degrees)		-148	-119	-60
R. Elbow- upper (degrees)		-13	-28	-20

5.3.1.1 Visual results

Visual comparisons for the three on-grid postures between posture prediction (exact posture) and predicted GRNN results are shown in Figure 5.2. In the figure, the red arrows refer to the target point locations. As mentioned, the GRNN results are expected to be accurate for these on-grid cases because they are some of the cases that the network was trained to predict. These chosen on-grid cases are evaluated to make sure that the network was designed well to have good prediction for all cases. As discussed in Chapter 3, the network should be tested for on-grid cases to make sure that the network was not trained to only predict the cases used to construct the network.

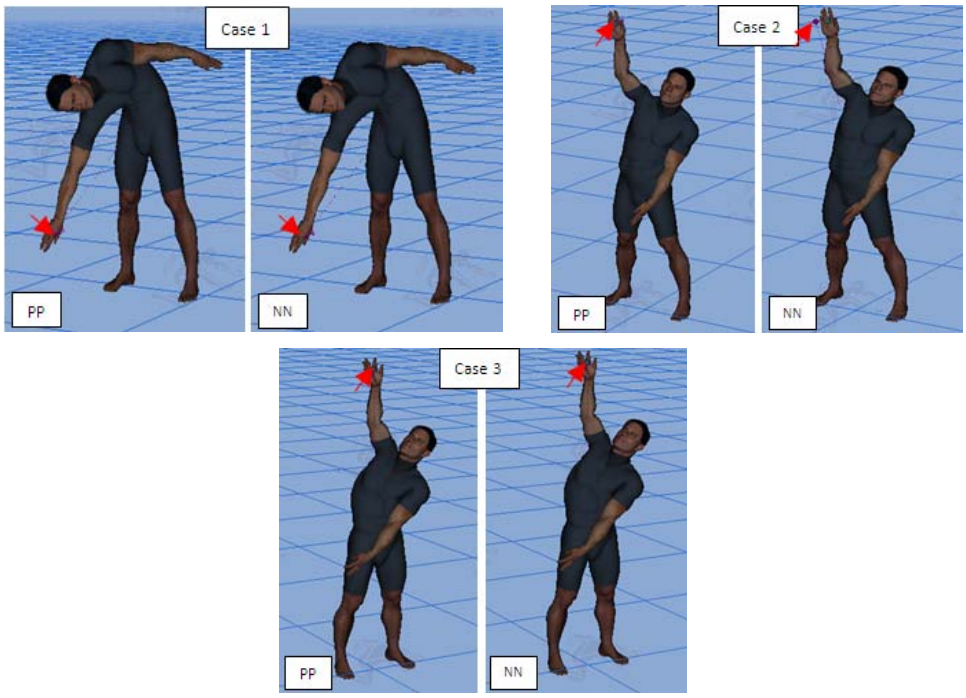

Figure 5.2: Three on-grid postures during the task of touching a point without external force for Santos posture prediction (PP) and GRNN (NN).

The visual results in Figure 5.2 show that the GRNN output postures failed to touch the target points exactly. The target point in each case is represented as a red ball, which is located by a red arrow. Note that there are some small errors in GRNN prediction for the joint angles, which lead the hand away from the required exact point location. The error was because the network is a general regression type, which interpolates between training cases, and it was not trained to predict the input target point with 100% accuracy. The network was trained to predict joint angles where even minimal errors in the predicted joint angles can manifest themselves as significant errors in the space (x, y, and z space). In all cases, Santos's hand moves toward the proper direction and close to the target point but with a small error.

This resultant error in touching the exact point was not an issue in the motion tasks presented in Chapter 4 because the tasks are totally different. In the touching point task, the target point position is an input for the network. The collected training cases

cover limited points in the grid space, where the other points are still predicted but with some error. The outputs in this task represent the joint angles to reach the input point, and predicting these values with minimal errors produces significant error in touching the exact point. On the other hand, the motion task inputs were not values that change the behavior of the predicted outputs totally, like target position. Regarding the motion task outputs, the predicted motion profiles have the same trend, where they change slightly depending on the inputs. Therefore, the created grid space in the motion task has fewer empty spaces than in the posture task. The predicted outputs in the motion task also have less variation in the general behavior of the predicted task outputs. Hence, the resulting errors in the presented on-grid results indicate that there are two potential options to overcome the problem of the accuracy in touching the point. These options are presented in the following:

1. Adding constraints to the network construction to force tuning the predicted postures from the network to be exactly on the proper position**.**
2. Collecting more training cases to cover more points in the reachable zones. Collecting more training cases improves the accuracy of the outputs. Although the obtained errors in this section were for the on-grid cases, constructing the network using more training cases will increase the network prediction ability.

Now, off-grid testing cases are shown in Figure 5.3, where red arrows refer to the target point locations. The figure visually compares three cases in which Santos was able to reach them without getting into any strange postures and with almost the same accuracy that was achieved for the on-grid cases. Even though the error of touching the target point is clearer in these cases than in on-grid cases, the network was still able to predict all 41 DOFs properly. The results show that the predicted joint angles, including the head and neck, were all tuned with the body corresponding to the target point position. The error in predicting the joint angles was minor, but still has a clear effect on touching the target point.

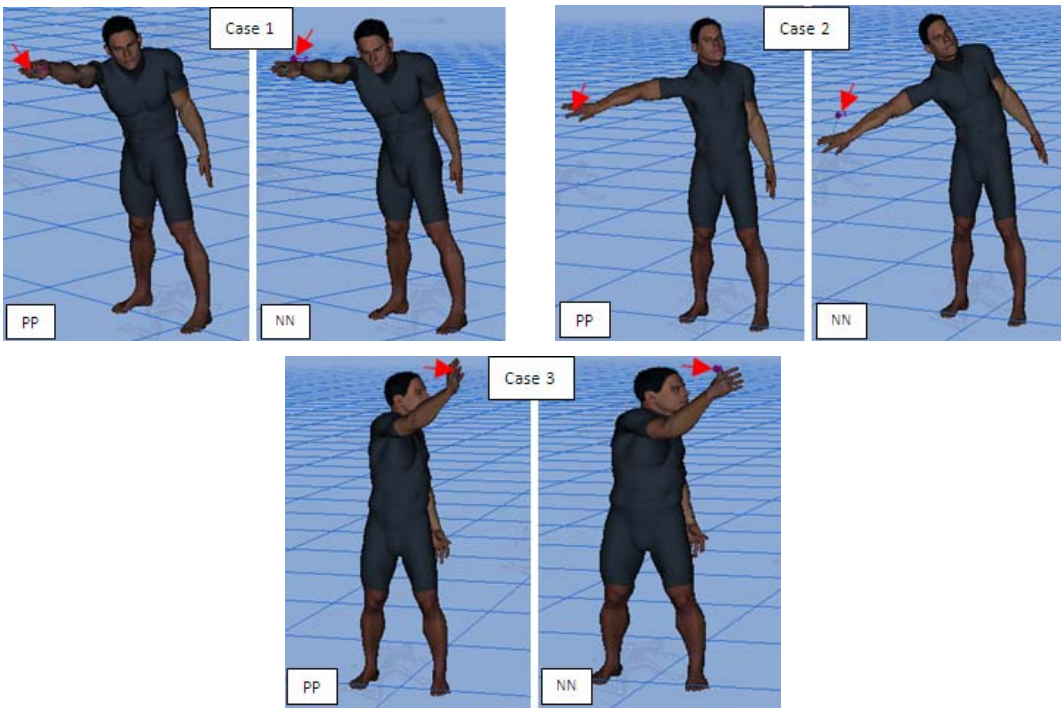

Figure 5.3: Three off-grid postures in the task of touching a point without external force for Santos posture prediction (PP) and GRNN (NN).

Like on-grid results, the accuracy issues exist in off-grid cases. The network ability to predict off-grid cases is similar to the on-grid cases for this task. Hence, the proposed options above to solve the accuracy problem could solve the accuracy of off-grid points too. Generally, incorporating the GRNN in posture prediction could enhance the prediction ability to eventually have fast, realistic, and task-based prediction of any point in the human reachable space.

5.3.1.2 Joint angles

This section presents the accuracy of predicted joint angles from the GRNN. Adjusted R-squares are plotted for the three on-grid cases in Figure 5.4. These plots are for joint angles resulting from the network and actual Santos PP outputs. It can be seen that the accuracy of the predicted joint angles is above 0.99 for all cases. Case 3 has a complete match between the exact and predicted postures (posture prediction and

predicted posture from the network). This accuracy was also shown in the visual results, where Santos was able to touch the point exactly. As discussed, it is expected to have a complete match when predicting some of the on-grid cases. These plots indicated that the network was able to interpolate all body joint angles properly to get acceptable accuracy; however, very small changes in some outputs still might lead to posture with the hand away from the exact target point. On the other hand, these results were statistically promising, since the network was able to predict all joint angles quickly and accurately.

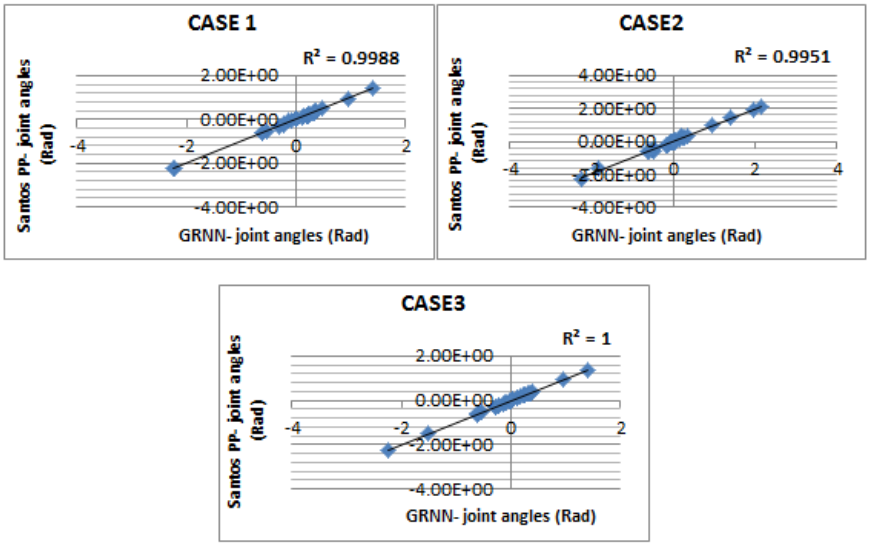

Figure 5.4: R-square values for GRNN vs. Santos posture prediction for 41 DOFs of three on-grid testing cases.

Similar to the on-grid cases, Figure 5.5 shows adjusted R-square values for the three off-grid testing cases. Even though those cases were not trained on the network, the results showed high matching between predicted values and the actual ones. The R-square values for all cases were above 0.97, which were highly accepted numbers in statistics. Hence, it was found that there is a similarity between off-grid target point predictions and on-grid ones. In conclusion, the network handled the task properly by providing postures close to the exact ones.

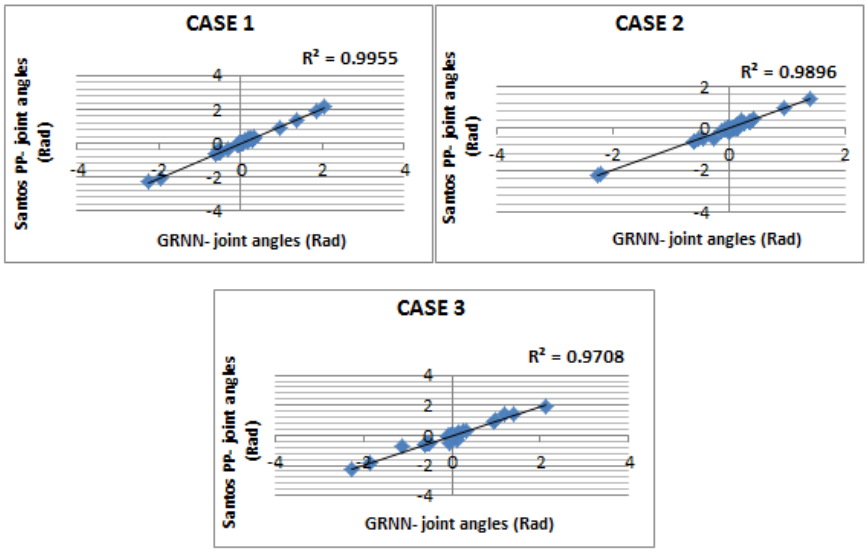

Figure 5.5: Adjusted R-square values for GRNN and Santos posture prediction for 41 DOFs for three off-grid testing cases.

Studying PP for touching a point using ANN showed that there is potential for the use of ANN to quickly predict realistic postures. This prediction could be used widely and in a task-based manner because, after training was completed, the GRNN was able to smoothly predict even off-grid points. In addition, it used a relatively small number of training cases, which could come from any other source, such as motion capture systems, since requiring a large number of training cases is not a problem for the GRNN type of ANNs. However, obtaining R-square values in posture prediction similar to those obtained for motion prediction in Chapter 4 suggests the potential for an issue of contact problem possibility for motion prediction as well as posture.

5.3.2 Posture prediction for touching point with external load

Results of posture prediction for touching a point with an external load are presented in this section. This task technically was an expansion of what was done in the first task. The GRNN was applied to include more conditions and predict more parameters. So, the network was trained to predict joint angles and torques. The task used

the same input parameters that were used for the first task (task of touching point without external force). The input parameter training values, however, differ from those used in the first task. The input values for the presented testing cases in this section are different from those presented in the first task too. Both on- and off-grid cases' results are presented in Table 5.7. Like the first task, those cases are compared visually and statistically.

Table 5.7: Input parameters for the two on- and off-grid testing cases in posture prediction for touching a point with an external load.

Input parameter	On-grid cases		Off-grid cases	
	Case 1	Case 2	Case 1	Case 2
Target position (cm): X	17	-49	-7	12
Y	-11	-30	40	-2
Z	-34	-49	-60	-66
R. Shoulder1- lower (degrees)	-19	-19	-10	-12
R. Shoulder1- upper (degrees)	111	111	80	90
R. Shoulder2- lower (degrees)	-23	-5	-15	-9
R. Shoulder2- upper (degrees)	123.5	50	90	70
R. Elbow- lower (degrees)	-148.5	-148.5	-100	-80
R. Elbow- upper (degrees)	-12.5	-12.5	-20	-30

This section has three subsections that separately discuss visual and statistical results for on- and off-grid testing cases. The differences between the results of this task and the previous one include: 1) this task has two testing cases for each of the on- and off-grid testing cases, and 2) the outputs of this task include joint torques in addition to joint angles.

5.3.2.1 Visual results

These results are obtained by running the posture prediction in Santos, under the exact mentioned conditions, for the exact results and the GRNN for predicted results. The joint angle values are performed at Santos's body and compared subjectively for both on- and off-grid cases. Figures 5.6 and 5.7 show the visual results for all testing points. The green arrow on Santos's hand shows the 100 N load that is applied on his hand in this task, while the red ones refer to the target points locations. Figure 5.6 compares the visual results of the two on-grid testing cases between the posture prediction and predicted posture from the network.

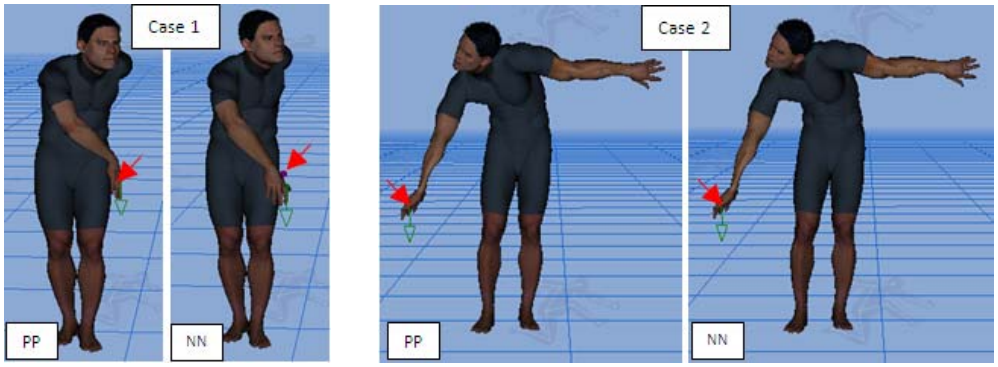

Figure 5.6: Two on-grid postures for the task of touching a point with external force for Santos posture prediction (PP) and GRNN (NN).

The above figure shows that the GRNN had accepted prediction for those on-grid points, training points; they were even better than those in the previous section. That might be a result of having more training cases for this network and/or the randomly selected target point positions that were used in the comparisons at both tasks. By chance, this task has more training cases than the first one, because the cases were collected randomly to cover all reachable zones. After collecting all training cases, there were more obtained cases for this task. It looks like Case 1 had more error in touching the target point than Case 2, which was very accurate. In both cases, the network had very good

results in terms of fast prediction; accuracy could be improved by having more training cases.

Figure 5.7 shows the visual results for the two off-grid testing points in which the network had the same trend in predicting previous points. In the figure, the red arrows refer to the target point. Santos had relatively large error in touching the target point in Case 1, while Case 2 was accurate. Generally, the off-grid testing cases had similar accuracy and prediction behaviors to the on-grid testing ones. Obtaining the same level of accuracy for on- and off-grid testing cases means that the heuristic method of determining GW in Chapter 3 works well for different tasks. Thus, the network predicted results for any point is comparable between both on- and off-grid points. Improving the accuracy of this task could be achieved by applying one of the potential options that were presented to solve the task of touching a point without external force.

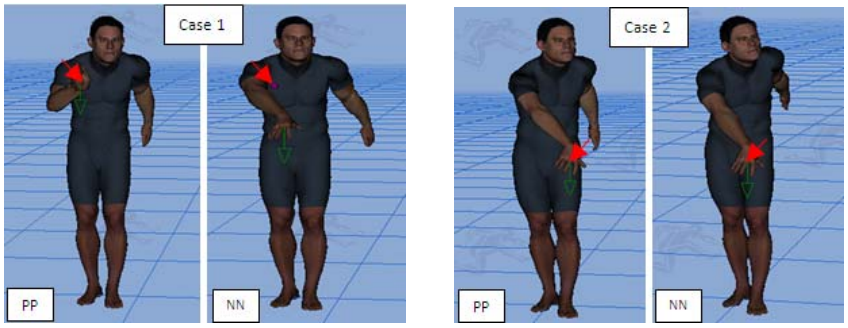

Figure 5.7: Two off-grid postures in the task of touching a point with external force for Santos posture prediction (PP) and GRNN (NN).

Generally speaking, it was found from these results that system prediction depends on: 1) the input combination (i.e., how close the point is to the training cases), 2) the number of training cases, and 3) the network properties, which are automatically selected in this study to have maximum accuracy for such a task.

5.3.2.2 Joint angles

In this and the following part of the results discussion, the two types of outputs (joint angles and torques) are presented and compared statistically by measuring adjusted R-square for all testing cases. The joint angle results are presented first in this subsection followed by joint torque results. The adjusted R-square plots are presented for the two on-grid testing cases in Figure 5.8. Case 2 had R-square equal to 1, while it was around 0.94 for Case 1. It is expected to have some on-grid points with 100% accuracy like Case 2 in the previous figure. These values match the visual results, which showed more accurate results for Case 2 than for Case 1.

These results were obtained because Case 2 was closer to the training cases than Case 1, which produces more accurate predicted results for Case 2. Generally speaking, the accuracy of predicting such a point increases if the point becomes closer to more training points. In other words, the accuracy of predicting one on-grid case could be lower than that of one off-grid case, because the on-grid point is further from the other on-grid cases than the off-grid case (i.e., the on-grid case is an extreme grid point).

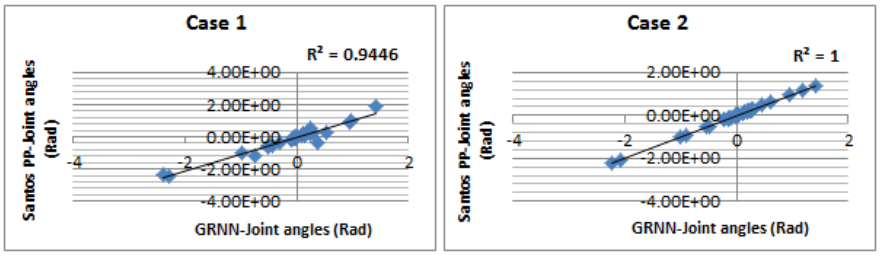

Figure 5.8: Adjusted R-square values between GRNN and Santos PP for 41 DOFs of two on-grid testing cases.

The second results to compare are the off-grid testing cases. Figure 5.9 shows the adjusted R-square values for both off-grid cases. The R-square values were 0.86 and 0.97

for Case 1 and Case 2, respectively. These results were shown visually in Figure 5.7 where Santos's hand was farther from the point in Case 1 than in Case 2.

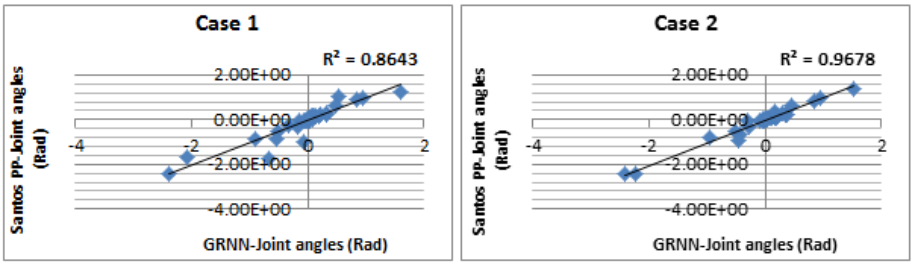

Figure 5.9: Adjusted R-square values between GRNN and Santos posture prediction for 41 DOFs of two off-grid testing cases.

In the visual results of Case 2 in Figure 5.7, Santos touches the target point exactly. That accurate result was obtained with R-square around 0.97, which indicates that visual results could be accurate even with R-square less than 1. That was shown in motion prediction applications when R-square values were less than 1, and were visually accurate. On the other hand, the first task had a test case with R-square values above 0.97, but also had more prediction error than this case, Case 2.

It is concluded that there was some visual accuracy indication in finding the R-square values, but not for all cases. Some off-grid points also have better results than on-grid points, even though the network was trained to predict those on-grid points. In general, higher R-square value is directly related to producing better visual results.

5.3.2.3 Joint torques

The third set of results in this section represents the second type of outputs, which are the joint torque values. The reason for including torques in the outputs was to examine the accuracy of the network prediction when having two different types of outputs. In this task, the outputs are joint angles and torques. The joint torque results with their adjusted R-square values for on-grid tasting cases are shown in Figure 5.10. Case 2

has R-square value equals to 1, which was the same value for the joint angle results of the same case, shown in Figure 5.8. The joint torque results of the on-grid cases had accuracy similar to that achieved in the joint angle results.

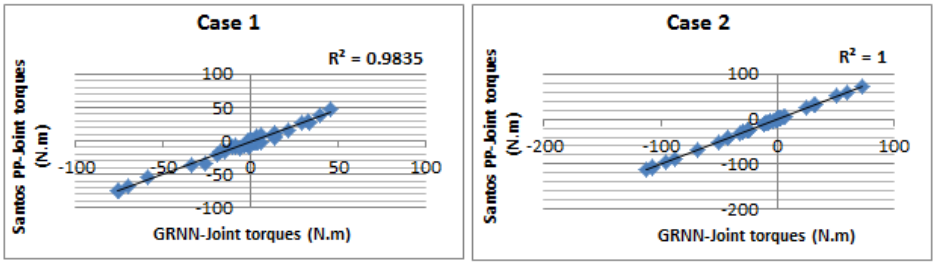

Figure 5.10: Adjusted R-square values for all on-grid testing cases. In each case, the R-square is plotted for all 47 DOF torques.

In Figure 5.11, joint torque plots with adjusted R-square values are shown for the two off-grid testing cases. Case 1 shows the lowest R-square value (0.97), which was the case with the lowest R-square value in joint angle prediction among all testing points. Case 2 has a highly accurate value of 0.99, where the visual results for this case was accurate too.

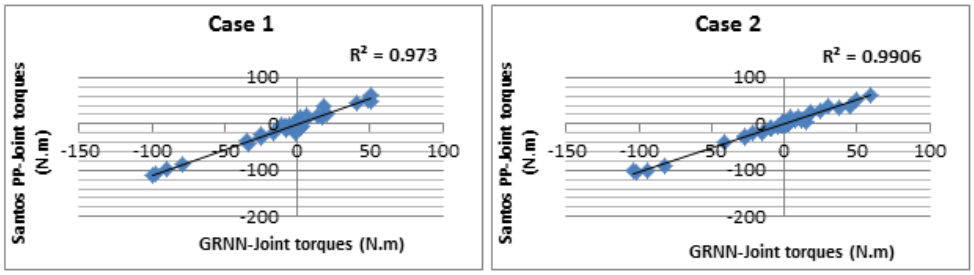

Figure 5.11: Adjusted R-square values for the two off-grid testing cases. In each case, the R-square is plotted for all 47 DOF torques.

Generally, the joint torque results were very accurate. The R-square values were above 0.98 for both on-grid cases, while they were above 0.97 in off-grid cases. For each of these on- and off-grid testing cases, the torque result (R-square value) is similar to that obtained in the joint angles result for that case. It is concluded that the network was able to handle and predict both types of outputs successfully. Both outputs were predicted with the same level of accuracy for each fed case.

This section indicated that one GRNN could be able to handle two types of outputs with the same success rate. Both visual and statistical comparisons were helpful in providing results indicating the accuracy of the GRNN in predicting human postures. Moreover, the results provide a slight indication of some relationship between angle and torque values at the same joint. This is because the joint torque changes when changing the joint angle. The significant indication for extracting this relationship from the predicted GRNN results is that the network could not only predict these outputs separately, but it also finds some relationships between them. Thus, the ability of finding this relationship might be the key to further analysis on the network properties (layers and neurons) to obtain clear information on the general human performance when performing a posture task, or to further study the use of other input parameters that might have more effect on the predicted outputs.

5.4. Discussion

Another application for ANN in DHM, specifically the GRNN type, was described in this chapter. This application is posture prediction, which has already been studied in the literature using many approaches, including some ANN types. On the other hand, the GRNN type of ANN has never been used for this purpose. GRNN has ability superior to that of other types of ANN in predicting a large number of outputs accurately. Thus, GRNN was used in this chapter to predict postures for two tasks. The tasks are touching a point on the front side of the body with and without external force applied on the hand. Since there is infinite number of points in front of the body, the GRNN was

able to use a relatively small number of training cases to produce generally accurate, fast, and promising postures for both tasks.

This study showed that the network's GW, which was determined automatically using the new approach, was varied depending on the applications and was successful. The GW values in both tasks were similar, 0.25 and 0.3 for the first and second tasks, respectively. This similarity in the obtained GW values concludes that having two different types of outputs for a task (like the task of touching a point with external force) does not sharply change the GW value from that obtained for a task with only one type of output (like the task of touching a point without external force). In this chapter, both performed tasks have the same inputs, but a very different number of outputs. Therefore, predicting two different types of outputs is not the main reason for reducing the accuracy of the network prediction capability. The task of touching a point without external force has only one type of output, but has relatively the same accuracy that resulted from the task of touching point with external force.

The contact problem (touching exact point or location) in this chapter was serious. Santos failed in exactly touching the target point for most of the tested cases, even the on-grid cases. This problem did not occur in the jumping up on a box task in Chapter 4, because the task had a smaller number of inputs and fewer gaps between training cases. Consequently, the GW value was very small, equal to 0.05, which produced accurate results from the network. Generally, to solve the contact problem in posture prediction tasks, there are two options:

1. Collecting many training cases to decrease the gaps in the training grid. Then, the new methodology for determining best GW will be able to set small GW and predict the task more accurately.
2. Adding constraints to the network construction to force the predicted postures from the network to be exactly in the proper position**.**

It is evident from this chapter that using an adjusted R-square value is relatively helpful for comparing results but that visual error could exist with a very high accuracy percentage. On the other hand, the results could have a relatively low R-square value with good visual results depending on the significance of the joints that have errors in their prediction. Some joints with minimal errors produce significant visual error in touching the exact point, while others do not produce that error (like joints that are responsible for the rotation).

Although many types of ANNs were used in studying posture prediction, studying two different tasks' posture predictions using the same ANN type resulted in some interesting conclusions about how to generalize posture prediction problems. This prediction could be done realistically and in a task-based manner once we know the general trend for the human when performing posture prediction, which is done by studying more and various posture tasks. Hence, the network could be used as an initial point for an optimization problem for the quickest and best posture prediction for a specific task. In addition, the network used a relatively small number of training cases, which could come from other sources such as motion capture systems. Eventually, generalizing posture prediction capabilities would facilitate a thorough understanding of human performance.

Generally speaking, it was found from the applications in this thesis that system prediction depends on: 1) the input combination (i.e., how close to the point the training case is), 2) the number of training cases, and 3) the network properties, which are automatically selected in this study to have the maximum accuracy for such a task. Moreover, the type of inputs and outputs that form a task could be studied to improve the performance of the network used. For example, training the network to predict joint center locations instead of joint angles has a prospective success in producing more accurate results in posture prediction tasks.

Along with the promising use of GRNN in posture prediction, there are some challenges and limitations in its current use that need to be addressed in future work. First, the accuracy of touching the target point was a problem when using GRNN even when predicting on-grid points. Second, the proper number of training cases to be chosen for such a task needs to be addressed. Two different numbers of training cases in which the corresponding accuracies were different were used for this application. Hence, this number should be optimized depending on: 1) the needed accuracy, 2) the task type, and 3) the number of inputs and outputs. On the other hand, using the GRNN in this chapter achieved the goal of studying initial work quickly, directing Santos to his proper posture.

CHAPTER VI

DECISION ENGINE FOR HUMAN PERFORMANCE MEASURES

6.1. Introduction

The successful use of artificial neural networks (ANN) in different digital human modeling (DHM) problems encourages examining the prediction of human performance measures (PMs) using a general regression neural network (GRNN) in this research. The PMs are functions that are minimized together in a posture prediction problem, as shown in the optimization formula in Equation 6.1. The difference between this application and the applications presented in Chapters 4 and 5 is that the literature has never predicted PM weights using ANN or any other method.

Find: Joint angles (6.1)

To minimize:

$$F(q) = w_1 f_{Discomfort} + w_2 f_{Joint\ Displacement} + w_3 f_{Max.Joint-torque} + w_4 f_{Total\ Joint-torque}$$

Subject to: Distance between finger-tip (end effector) and target point = very small value.

Human PMs are considered the core of human performance drivers, which direct the internal decisions and behaviors of humans while they are performing various tasks. Humans make decisions about which PM to use and minimize, and behaviors are the results of using specific PMs. If a specific method could realistically predict these PMs, it would be a significant addition to DHM applications, allowing imitation of the human brain. However, the main challenge in studying human PMs is that there is no direct method for directly predicting PMs in any DHM application.

In human posture prediction, PMs directly influence posture results because the human tends to minimize one or more PMs when performing a task, producing different results. This study uses four joint-based PMs (discomfort, joint displacement, maximum joint torque, and total joint torques), and this chapter presents a new strategy for

predicting the weights $[w_1, w_2, w_3, w_4]$ of these four PMs using GRNN (see the cost function in Equation 6.1).

This chapter also examines the applicability of two approaches for directly and indirectly collecting training cases (i.e., weight values) for the applied network. The first training approach depends on direct extraction of the PM weight values manually from predicted postures. The PM weights in the training cases are chosen subjectively based on posture prediction experiments in which the weights are changed until the posture is visually accepted for the intended task. The second training approach uses motion-captured postures for indirect extraction of the PM weights. This method is more innovative and reliable for predicting PMs because it depends on extracting the PM weights indirectly from motion capture (Mo-cap) postures. The method basically extracts the PM weights by solving optimization problem for provided Mo-cap posture. The both proposed approaches for collecting the training cases will be described in details in the following sections. In summary, the following contributions are provided in this chapter:

1. Developing a decision engine for determining and selecting human PM weights using subjectively selected postures as training cases.
2. Proposing a mathematical formulation for indirectly extracting human PM weights from motion-captured postures. The proposed method is validated using one training case. A training case was extracted (i.e., the PM weights were extracted) from recorded Mo-cap data with an optimization algorithm.

6.2. Training with Predicted Postures

Posture prediction using multi-objective optimization (Equation 6.1) in Santos is well developed to be used as a source for training the proposed network to predict the four PMs for various tasks, including those studied in Chapter 5. Training cases in this method are collected manually using posture prediction in Santos and using the four PMs together to be minimized in touching a point with and without external force on the right hand.

Training cases were collected manually to have different PM combinations for touching different points in front. The collected training cases were subjectively accepted based on the visual acceptance of the produced posture. Same cases were collected with the same inputs except with the existence of a 100 N external force on the right hand for the second task. Table 6.1 shows the input and output ranges and values for the training cases. The number of input parameters and outputs were the same for both tasks. In the table, there are nine inputs, including three for the target position (X, Y, and Z) and six for the upper and lower limits of the right shoulder flexion-extension (R. Shoulder1), shoulder adduction-abduction (R. Shoulder2), and elbow flexion-extension (R. Elbow). The table is arranged so that the maximum and minimum columns represent the maximum and minimum training values used in the training cases for the corresponding row (parameter). Input parameters with Value 1 and Value 2 have only two fixed values in the training cases. So, the parameter has either one of those two values in the collected training cases. For example, target position is expressed in three-dimensional values (X, Y, and Z), where each training case has a value within the range between the minimum and maximum values in the three dimensions. Joint ranges of motion (ROMs), however, have two fixed values that are used in the training cases. Each training case uses only value 1 or value 2 for all input ROMs.

Table 6.1: The training values for the input and output parameters for the proposed decision engine. Both tasks are included and have the same training inputs but different outputs.

	Parameter name	Value 1	Value 2	Minimum	Maximum
Inputs	Target Position: X	---	---	-97	52
	Y	---	---	-50	103
	Z	---	---	-102	1
	R. Shoulder1-L	-23	-5	---	---
	R. Shoulder1-U	123.5	60	---	---
	R. Shoulder2-L	-19	-5	---	---
	R. Shoulder2-U	111	50	---	---
	R. Elbow-L	-148.5	-80	---	---
	R. Elbow-U	-12.5	-40	--	---
Outputs (No Force)	Discomfort	---	---	0.05	0.4
	J. Displacement	---	---	0.8	0.15
	Max. J. Torque	---	---	0.05	0.5
	Total. J. Torques	---	---	0	0.25
Outputs (with Force)	Discomfort	---	---	0.1	0.4
	J. Displacement	---	---	0.2	0.6
	Max. J. Torque	---	---	0.1	0.4
	Total J. Torques	---	---	0.05	0.3

The input training values also were exactly the same for both tasks (tasks of touching point with and without external load), while the output PM weights at the training cases were different because of the force existence. The outputs included four parameters representing the four PMs that were used in in this study, and they differ according to task (i.e., they have different colors in the table). These PMs are: discomfort,

joint displacement, maximum joint torque, and total joint torque. The input combinations for all training cases in both tasks are shown in Tables B.5 and B.6 (Appendix B).

For each collected training case, the four PM weights were changed until Santos touches the point properly and his posture looks realistic. The sum of the PM weights in each training case equals 1. Since humans behave differently, the collected training postures and the PM weights were considered accepted based on what we found accepted for the given points and the specified tasks. After collecting the training cases, the network was constructed and trained using those cases. There were 21 training cases; each had nine inputs and four outputs. Constructing the network in this application was slightly different from the previous applications. The new strategy of determining Gaussian width (GW) was not applied in this application for the following reasons:

1. Initially, some off- grid cases were collected to determine the best GW for this application. However, the best GW value was 2, which was very large, and the best R-square was very low, around 0.2, using this GW value. This GW value provided high error in the general prediction because it was large. That GW value was obtained because the network was not able to produce any value for the testing cases without having a large GW value to produce outputs from many neurons at the same time. In general, if the inputs are normalized, it is impossible for the GW to equal 2 for any task.
2. The application in this chapter is different from the ones in Chapters 4 and 5. Here, the network predicts the general trend in selecting the combination of the PM weights in the training grid. Hence, generalization of the prediction of output weights is needed more than prediction of the exact outputs for testing or training cases (i.e., there is no need to have a very low GW value for exact prediction).

Therefore, it is better to have a reasonable GW value that provides an accepted weight combination for each zone or area in the training grid. Producing the PM weights from the network with some error will not change the general trend in the predicted

postures, because the PMs are summed together in the optimization problem (Equation 6.1). Using a very small GW value produced accepted predictions for the on-grid training point only, and failed in general prediction of accepted weight values for each region on the grid. On the other hand, a large GW value produces similar outputs for any point at any region in the space, and fails to produce accepted outputs at different grid zones (i.e., the network produces the same outputs for different inputs). Thus, the GW was chosen around the middle of the normalized input value and equal to 0.4. The best GW value was found to be around the middle of the input values in previous literature (as shown in Chapter 3). In addition, all previous applications in this thesis had GW values of less than 0.5, which indicated that the GW should not even reach the middle of the maximum input normalized value. So, both tasks, touching a point with and without external load, use a GW equal to 0.4 and 21 training cases.

6.2.1 Results

This section presents the results from the trained network to predict the PM weights. Expressing the results of this application is slightly different from expressing those from previous applications in this thesis, because training cases were collected on subjective bases. Hence, test cases results are evaluated based on the visual appearance of the postures that are produced by using the four predicted PM weights. Both tasks are evaluated in this section for on- and off-grid test cases, each with three test cases.

6.2.1.1 Off-grid results

For both tasks, we evaluate their networks using the same testing cases, because they used the same training input values. Table 6.2 shows the input parameter values of three off-grid test cases. These same cases are used to test the predicted results from both tasks' networks. In these cases, the target points had positions between the maximum and minimum training values, while the ROMs had values between the two values that were

used in training. The predicted outputs from GRNN, which are the PM weights, should be subjectively realistic.

Table 6.2: Input parameter values for three off-grid testing cases.

Input Parameter	Case 1	Case 2	Case 3
Target Position: X	-3	40	-37
Y	21	-13	94
Z	-52	-57	-37
R. Shoulder1-L	-10	-8	-17
R. Shoulder1-U	110	90	70
R. Shoulder2-L	-19	-10	-15
R. Shoulder2-U	59	80	100
R. Elbow-L	-120	-130	-90
R. Elbow-U	-20	-20	-35

The predicted outputs for the three off-grid testing cases are shown in Table 6.3, which includes the outputs for both tasks. The sum of the weights was 1 for all training sets. Similarly, the sum of the predicted weights in each case in the table is around 1. In general, the network predicted high output fraction for joint displacement PM in all cases, and for both tasks. The network also predicts relatively low fraction for the total joint torque PM. These results match the general trend for the weight values in the training cases (See Tables B.5 and B.6 in Appendix B).

Table 6.3: Predicted output weights from GRNN for the four PMs in three off-grid testing cases for both tasks.

Task type	Case #	Discomfort	Joint Displacement	Max. Joint Torque	Total Joint Torques
No force	**1**	0.11	0.71	0.07	0.1
	2	0.13	0.68	0.1	0.09
	3	0.21	0.55	0.19	0.04
With force	**1**	0.12	0.53	0.21	0.15
	2	0.14	0.48	0.24	0.13
	3	0.19	0.42	0.24	0.14

The weight values in the table show that there were some differences in the predicted PM weights between the two tasks, with and without external force tasks. In general, the tasks have different predicted weight values when touching the same testing point. Joint displacement PM was the dominant PM in both tasks, but with a smaller fraction in the second task. Maximum joint torque and total joint torques were more important in the case of external force existence, because they had larger weight values in the second task.

For the first task, comparing the resulting postures when using the predicted PM weights was more useful for visually examining the efficiency of using GRNN and the predicted numbers. The visual results were obtained by asking Santos posture prediction to do the tasks using the GRNN predicted weights. Figure 6.1 shows the visual postures for the three off-grid testing cases for the first task when using the GRNN-predicted PM weights. Target points in the cases are represented with small red balls. The postures were visually accepted and relatively realistic for Santos. He was able to reach and touch the point in all cases properly and without bad movement or starting posture. Generally,

these postures look better than those produced when using one or two PMs, because predicting postures using four PMs is more realistic and accepted under various input points and conditions like load existence.

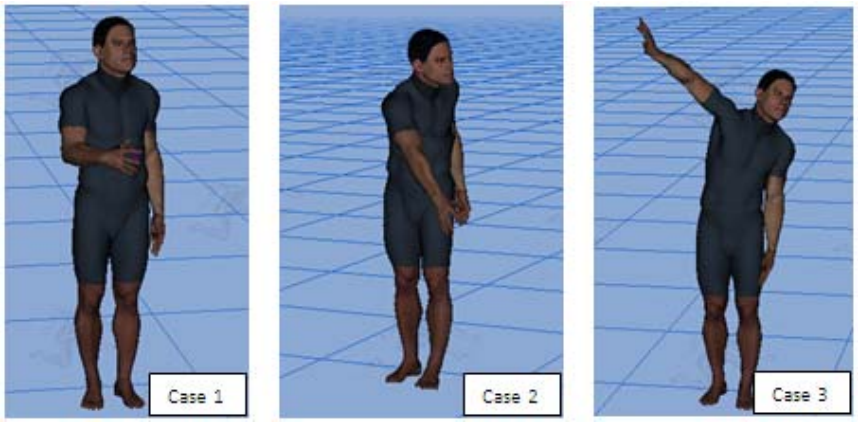

Figure 6.1: The produced postures for three off-grid test cases when using the GRNN-predicted PM weights in the task of touching a point.

Regarding the second task, Santos reached the same points but with external force on his right hand. The differences in the resulted PM weights between this task and the first task (the task of touching point without external load) were clear. Figure 6.2 shows the visual postures for the three off-grid testing cases at the second task when using the GRNN-predicted PM weights. The way Santos reached the same points was different because of the force on his hand. Santos's postures look different from those from the first task because of load existence. The results were still achieved successfully and realistically, considering the load effect on the produced postures. The green arrow in each posture in the figure represents the 100 N load on the right hand. However, Case 3 appears to be the least accepted posture among the tested postures, because it is close to the extreme values of the inputs in the training grid (i.e., it is the furthest testing case from the training points; see Table B.6 in Appendix B.

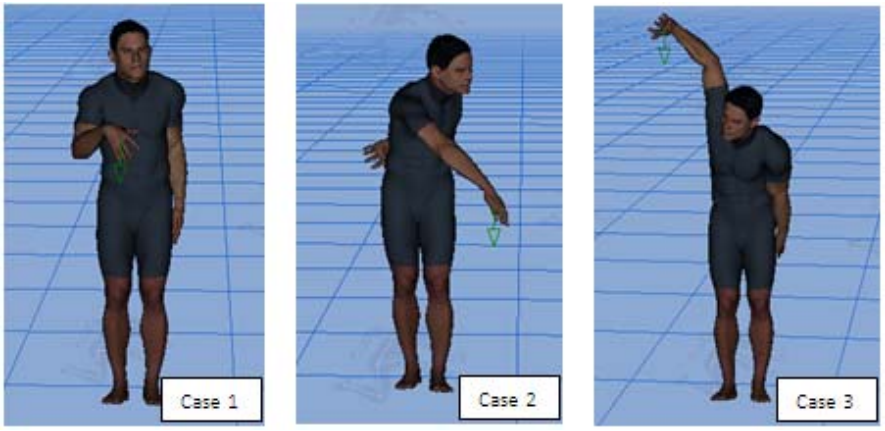

Figure 6.2: The produced postures for three off-grid test cases when using the GRNN-predicted PM weights in the task of touching a point with an external load.

Compared to the first task, Case 1 showed more bending in Santos's hand as a result of applying a load as well as more flexion in the elbow. This result is close to what a human normally does when he lifts a load. Similarly, Cases 2 and 3 showed the same effects on the produced postures, where different postures were obtained from those in the first task. In Case 2, Santos moves his shoulder closer to the point, which might be to decrease the torque on it while carrying that load. Case 3 was similar to the previous cases where his elbow bent more than it did in the first task. Hence, the second task presented and enhanced the idea of having some correlations between different PMs on a task-based manner, because the PM weight values in the training cases differed between the first and second tasks as well as the predicted results in the testing cases.

6.2.1.2 On-grid results

On-grid testing cases were also studied in this thesis to check the differences between the manually selected PM weights (actual training cases) and the predicted weights from the GRNN. It was important to study whether the network produces more realistic results than training cases, because the chosen training weights were selected

randomly during the training, which depended on and treated each case separately. Table 6.4 shows the three on-grid testing cases that were compared in this study.

Table 6.4: Input parameter values for three on-grid testing cases.

Input Parameter	Case 1	Case 2	Case 3
Target Position: X	-63	51	-56
Y	45	35	-2
Z	0	-15	-89
R. Shoulder1-L	-23	-23	-5
R. Shoulder1-U	123.5	123.5	60
R. Shoulder2-L	-19	-19	-5
R. Shoulder2-U	111	111	50
R. Elbow-L	-148.5	-148.5	-80
R. Elbow-U	-12.5	-12.5	-40

The resulting PM weights from the GRNN and the exact training outputs were compared. These results are presented in Table 6.5, which also presents the results for both tasks. The testing cases in the table are arranged so that the results of the first task are presented first. For the first task, the exact results are presented first, followed by the predicted results from GRNN. The same arrangement is followed for the second task in the same table.

Table 6.5: Three on-grid testing cases for output PM weights at both tasks. The results include the exact and predicted values.

Task type	Case #	Discomfort	Joint Displacement	Max. J. Torque	Total J. Torques
No force	**Exact: 1**	0.2	0.6	0.15	0.05
	2	0.1	0.25	0.4	0.25
	3	0.15	0.5	0.25	0.1
	GRNN: 1	0.16	0.52	0.23	0.09
	2	0.17	0.34	0.35	0.15
	3	0.16	0.49	0.23	0.07
With force	**Exact: 1**	0.25	0.5	0.1	0.15
	2	0.4	0.35	0.2	0.05
	3	0.3	0.3	0.3	0.1
	GRNN: 1	0.2	0.46	0.2	0.14
	2	0.26	0.37	0.25	0.13
	3	0.29	0.33	0.29	0.09

The GRNN provides some variations from the exact ones, which is expected because the GW was set in the middle to equal to 0.4. This value for GW does not provide exact results for on-grid prediction. However, having GW in the middle allows handling more variation or oscillation in the training cases, which was the case in this application, where the training case outputs were chosen randomly. The predicted results from GRNN were close to the exact values, and each predicted PM weight was confined within its training range.

As mentioned, these results in numbers did not provide much useful information about the degree of successfulness of the GRNN prediction. It is also important to remember that small changes in PM weights might lead to very different postures. Hence, visual results were compared by predicting postures using PM weights from the predicted GRNN and the exact training set. Figure 6.3 shows the visual postures for three on-grid testing cases in touching a point without a load on the hand. The cases are arranged so each case presents both predicted and exact postures.

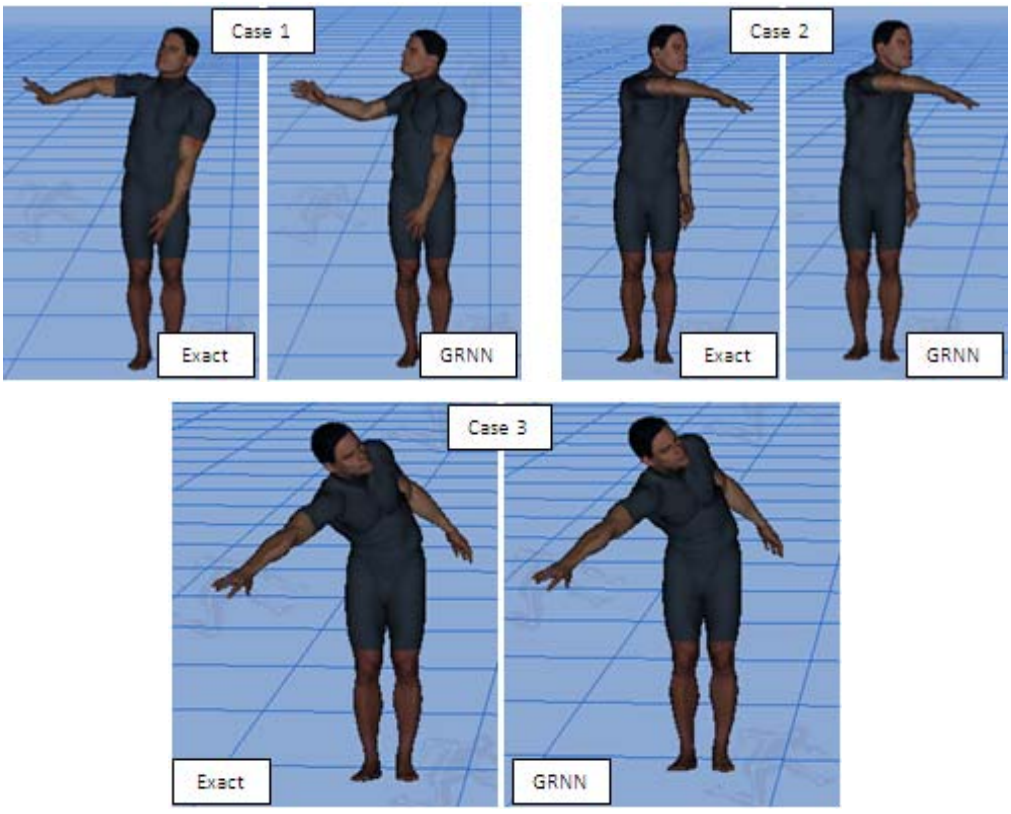

Figure 6.3: Three on-grid test posture cases for the actual (Exact) and predicted PM weights in the task of touching a point.

Since the training sets were subjectively selected, testing results are also compared subjectively based on the visual appearance of each posture. For this task, all

results in the above figure are accepted, where the GRNN on-grid postures were very close to those from the exact results even though the weight values were slightly different. Cases 2 and 3 in the figure show matching between the exact and predicted postures. The GRNN provided a different posture in Case 1, which would be considered at some point better than the exact training case. The reason for that is that the network predicted better weight combination than was chosen for that case during the training. In other words, the curve or prediction for the training cases that the GRNN produces might have better generalization for the weight predictions than selecting the weights randomly. Generally, all postures were accepted from the GRNN predicted weights as well as the exact ones. These results indicate that changing the produced posture solution sensitivity to the change in the PM values varies depending on the intended posture and task. Small changes in the predicted weight values in Case 1 produce a different optimization solution but did not change the results in Cases 2 and 3 for the exact and predicted results.

For the second task, touching a point with a load on the hand, the same three on-grid test cases were compared visually in Figure 6.4. The cases in the figure are for touching the same exact point except with existence of the load, indicated by the green arrows. Like the results of the first task, postures from the predicted PM weights were also slightly different from those in the exact training cases. In this task, two cases also matched. Cases 1 and 3 matched, and the predicted posture in Case 2 was different. In that case, Santos bent his elbow in both postures but in different rotations. This case was again subject to personal preference as far as deciding which resulting posture was better. In this task, all cases were also accepted in terms of someone carrying 100 N on his hand.

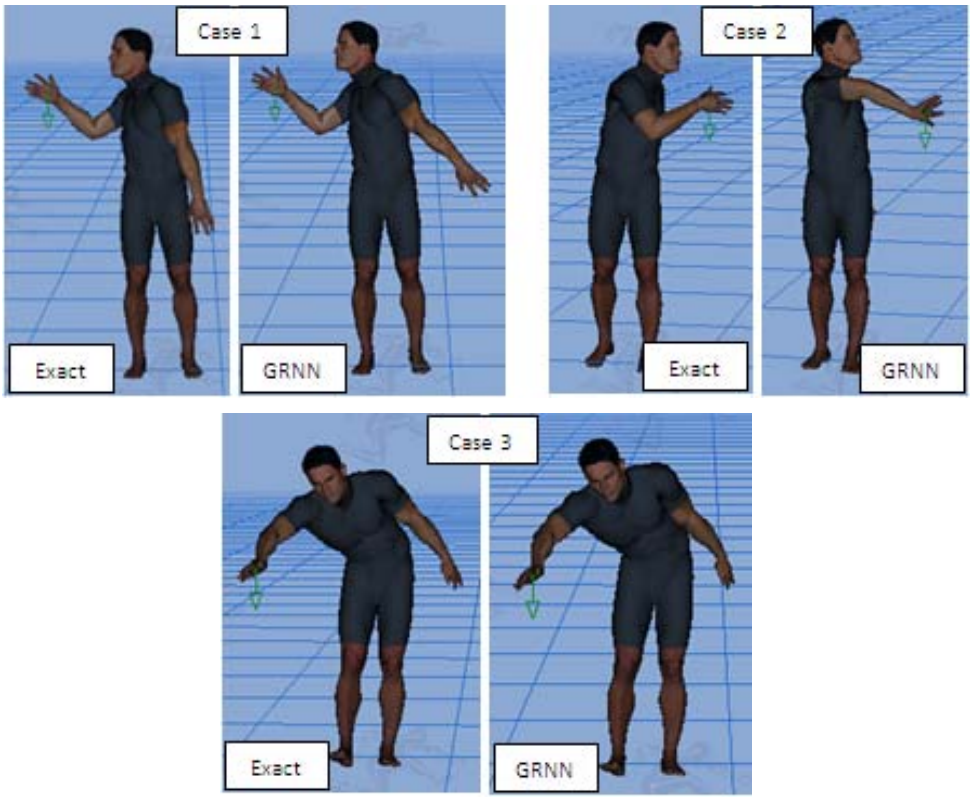

Figure 6.4: Three on-grid test posture cases for actual (Exact) and predicted PM weights in the task of touching a point with external force.

6.3. Training with Motion Capture (Mo-cap)

Since Mo-cap is a source for providing realistic data about human movement when doing a task, many scholars depend on Mo-cap to study human behavior. In this section, a new Mo-cap-based methodology is proposed to extract human PM weights that control human performance. Given enough Mo-cap data, this method should replace the method of training the network using manually predicted postures.

This method depends on solving an optimization problem (Equation 6.2). In this problem, the given data are five vectors of 55 DOFs with some other task-based input like target point position, load existence, sitting/standing, etc. The five given vectors represent one posture from Mo-cap posture and four postures from running Santos posture prediction, each using only one of the four PMs for the same task (i.e., touching

the same point with the same conditions). In the optimization problem, PM weights are the design variables that need to be optimized. The objective function to be minimized is the difference between the Mo-cap posture ($\tilde{\boldsymbol{q}}^M$) and the weighted sum of the four joint angle vectors [$\tilde{\mathbf{q}}_1, \tilde{\mathbf{q}}_2, \tilde{\mathbf{q}}_3, \dots, \tilde{\mathbf{q}}_w$], which results from running posture prediction using each of the four PMs. To normalize the weight values (i.e., make their total equal to 1) and limit them so that their optimized values are always positive, five constraints are added to the problem. The optimization problem is solved using sequential quadratic programming method by SNOPT software (Gill et al., 2002). After solving the optimization problem, the resulting posture from the four PMs combined in a weighted sum should be accepted, because it matches the actual Mo-cap posture.

$$
\begin{aligned}
&\text{Given:} && \tilde{\boldsymbol{q}}^M, \tilde{\boldsymbol{R}}, L, \text{ and additional task parameters} \\
&\text{Find:} && \tilde{\gamma}^w \\
&\text{To minimize:} && \left|\left|(\gamma_1\tilde{\mathbf{q}}^1 + \gamma_2\tilde{\mathbf{q}}^2 + \cdots + \gamma_w\tilde{\mathbf{q}}^w) - \tilde{\mathbf{q}}^M\right|\right| \\
&\text{Subject to:} && \textstyle\sum_{i=1}^{w} \tilde{\gamma}^i = 1 \\
& && 0 \le \gamma_1 \le 1 \\
& && 0 \le \gamma_2 \le 1 \\
& && 0 \le \gamma_3 \le 1 \\
& && 0 \le \gamma_4 \le 1
\end{aligned}
\tag{6.2}
$$

where: $\tilde{\boldsymbol{q}}^M$: Mo-cap joint angles (size: 1x55).
$\tilde{\boldsymbol{R}}$: target point position (x, y, and z) (size: 1x3).
L: external load value (size: 1x1).
[$\tilde{\mathbf{q}}_1, \tilde{\mathbf{q}}_2, \tilde{\mathbf{q}}_3, \dots, \tilde{\mathbf{q}}_w$]: joint angle values from different performance measures.
$\tilde{\gamma}^w = [\gamma_1, \gamma_2, \gamma_3, \dots]$: performance measure weights.
w=4 (number of PMs).

This training method could be populated to be used over many postures at various tasks. Given enough Mo-cap training data for such a task, this method will be able to train GRNN properly to predict that task. Further proper prediction for PM weights at different tasks under many conditions could lead to a thorough understanding of the

relationships between the PMs. Extracting these relationships will be a huge step toward understanding how and why people think as they do when accomplishing such a task.

6.3.1 Results

The previously described approach depended on subjective evaluation, training with predicted postures, in order to determine PM weights for training. A more precise approach, however, was needed to provide these weights for proper training. Thus, a new Mo-cap-based approach was introduced in Equation 6.2 for indirect extraction of the PM weights. In this section, we examine the validity of the proposed approach. One Mo-cap posture is collected to apply the new approach. Consequently, one training case is extracted and evaluated.

In the collected Mo-cap posture, all 55 DOFs are known. This Mo-cap posture, shown in Figure 6.5, is represented on Santos. The task was to touch a point in front with the right hand, similar to what was done in posture prediction applications in Chapter 5.

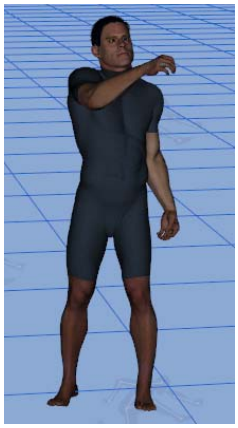

Figure 6.5: Motion capture posture on Santos.

Then, posture prediction was run four times on Santos to touch the same point using each one of the four PMs that were mentioned. Each resultant posture in Santos

posture prediction was obtained from minimizing only the specified PM. Figure 6.6 shows the resultant postures from Santos posture prediction when minimizing the discomfort, joint displacement, maximum joint torque, and total joint torques.

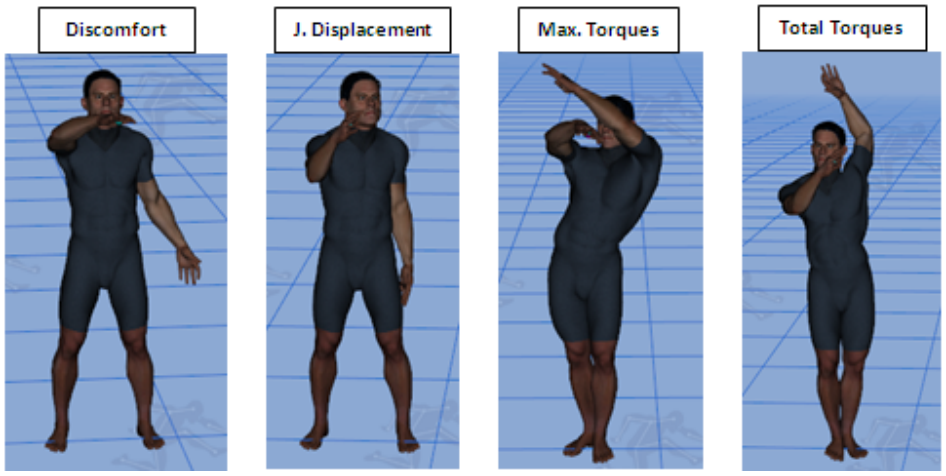

Figure 6.6: The four resultant postures for running Santos posture prediction with the four PMs.

The figure shows that none of these four postures was exactly like that from Mo-cap. However, the posture produced when minimizing joint displacement was the closest to the Mo-cap one, because this PM provides the most realistic posture for touching point in front side of the body. After obtaining the joint angles from these four postures, the optimization algorithm was run to find the optimal sum for these four postures that produces the exact or closest posture to the Mo-cap one (Equation 6.2). The algorithm finished successfully and provided the optimum weights. The weights were 0.28, 0.57, 0.07, and 0.08, which represent discomfort, joint displacement, maximum joint torque, and total joint torques, respectively. Each of the four postures had weight or contribution in the optimal solution, which means that they are all important in producing actual and realistic posture. After running Santos posture prediction using these PM weights, the

visual result is shown in Figure 6.7. This posture was the closest posture to the Mo-cap one.

Figure 6.7: The resulting postures for the optimal weight combination.

The result was visually better than all other postures that were produced using single PM except the one that produced from joint displacement. However, the joint displacement PM, as mentioned, cannot provide realistic postures when the point located behind the body or incase if external load is applied on the hand. Predicting the posture in Figure 6.7 was based on trying to make it as close as possible to realistic motion-captured posture, and it was close to that posture. The optimal posture, however, had some small differences from the exact one. The feet position and left hand were the main notable differences. Otherwise, the predicted posture was generally accepted. In addition, since optimization was used to produce this posture, it was the best achievable posture when the four PMs were used. In addition, the optimal weight values were similar to what was obtained in the first method. In that method, joint displacement weight value was generally dominant and had the largest value among all weights. Similarly, the optimal joint displacement weight in this test posture was dominant and equal to 0.57.

6.4. Discussion

The goal in this chapter was to understand how humans perform to complete a task by using ANN to predict some PMs that control this performance. Since no work was done to study and determine which performance measures drive humans to accomplish tasks, this chapter demonstrated using ANN as a decision engine for predicting the weights of human PMs. These PMs are functions that control human posture. The PM weights need to be determined to predict proper posture for such a task. The PMs are combined in one equation (Equation 6.1), where each PM has a weight that determines the importance of that PM in the problem.

Two new methodologies were presented in this chapter to provide a training source to use GRNN as a decision engine to predict four PM weights (discomfort, joint displacement, maximum joint torque, and total joint torques). These methods differed as far as the source of the collected training cases. The first method depended on setting the four PM weights subjectively to produce proper postures. The method was examined using two tasks: touching a point in front of the body using the right hand, and touching a point in front of the body with the right hand with a 100 N load on the hand. The second method depended on using Mo-cap postures to indirectly extract the four PM weights. The second proposed method is more promising, since it depends on the use of real postures to train the network. This research was the first to try to predict the human PMs and their importance in a task-based manner.

Since the training cases were collected subjectively based on visual acceptance, the results were evaluated visually too. The results showed that the GRNN predicted the PM weights well. The network predictions for different on- and off-grid test cases were visually accepted and comparable to those obtained manually. This method is helpful because it saves the time of trying many PM weight collections until the postures look better. Actually, some predicted PM weights were better than those used in the training, like Case 1 in the on-grid test results. In addition, studying two different tasks showed

how PM weights change according to the task to be achieved. So, this method was direct and easy to use for training the network to predict these PM weights for different tasks.

After applying both tasks in the first method, predicting weights for the task of touching a point in the front with and without force, general trends were drawn about the importance of each PM in those two tasks. It is interesting that people tend to minimize their joint torques in cases where there is applied load but minimize the movement (joint displacement) when no load exists. Even though the training cases in this method were obtained by manually predicted postures, the accepted network predictions for these PMs might be successfully generalized in future work to have more tasks and more conditions or inputs for the same task.

In the second proposed method, motion-captured postures were considered as a source of training. Since direct PM extraction from Mo-cap was impossible, an optimization-based formula was presented in this chapter to optimize and extract the PM weights from the given motion-captured joint angles. This method succeeded in extracting PM weights from Mo-cap data. Therefore, it is considered the first approach that obtains PM information from a reliable source like Mo-cap.

Unlike the first method in this chapter and the previous application in this thesis, GRNN was not used to apply the second method, because sufficient training data were not available. However, examining the validity of the proposed approach was more important at this level because once the initial test of this approach works well, collecting more training cases for the sake of using GRNN will be the default. Then, the results of this approach will be guaranteed because GRNN already showed high and promising success when it was used for other DHM applications in this research, including the first method proposed in this chapter. Future work should entail obtaining the PM weights from additional Mo-cap data using the proposed method in this chapter and then using the obtained weights to train the network.

Another validation for the second approach could be performed by comparing the weight values that were obtained from the optimization and those from manual selection in the first approach. The weight values in the first approach showed that the general dominant PM was joint displacement for the task of touching a point in front. This domination was similar to what the optimization produced, which was 0.57 or 57%. Another interesting conclusion was that the all optimized weight values in the second approach, which used actual Mo-cap posture, had values and contributed in the optimal solution. That means that they are all important in producing actual and realistic posture, and prediction of any posture should not be limited by using one or two PMs. Therefore, this conclusion enhanced the goal of this research when it referred to the need for predicting all PMs.

The work in this chapter was unique in terms of using Mo-cap data for the purpose of providing a source of training the GRNN to predict proper weights for different human PMs in a task-based manner. Eventually, GRNN could perform as a decision engine for human PMs that control posture prediction and many other DHM fields. In addition, the first approach proposed in this chapter could be used as extra subjective validation for the second approach when it is used in the future for other tasks.

CHAPTER VII

SUMMARY AND CONCLUSIONS

7.1. Summary

The DHM world has expanded rapidly, but the current human models still need to be more realistic in terms of being predictive for the intended task and responsive to different conditions during the task. This thesis investigated the use of artificial neural network (ANN) to solve problems with predicting and studying digital human model (DHM) like the speed of calculations in motion prediction problems. It also aimed to predict the human performance and mimic the drivers (performance measures, PMs) that control task performance.

Some DHM fields, like motion prediction, have mature models that have been validated successfully for predicting motions of different tasks. Such models, however, are need relatively long time to run and solve motion problems like predictive dynamics (PD) (Xiang et al., 2008), and thus they cannot be incorporated in real-time motion prediction. Other models are limited in providing realistic prediction for various task conditions like those come from prerecorded motions.

Another key limitation in predicting various DHM problems is in finding the best human PMs, which are the functions that a human tries to minimize when performing any posture, motion, or other task. Currently, these PMs are mainly used for posture prediction problems, the formulation of which was described in Chapter 5 in Equations 5.1 to 5.3. Generalizing the use of proper PMs for any task and problem will be a huge step toward understanding human performance, and achieving successful prediction for the PM combinations in various DHM problems will lead to improvements throughout the DHM field. Until now, no work has been performed toward direct selection or prediction of various PMs in any DHM field. Thus, the above issues should be solved if

we want to perform better and more realistic human performance prediction in a task-based manner and, eventually, understand what drives human performance.

ANN was included in this thesis to solve the provided DHM problems, because it had previously shown an excellent ability to predict any practical system and because it produces real-time outputs for the system when any of the system inputs are changed. Important work has been done to predict various DHM applications using various approaches, including those incorporating different types of ANN, but nobody had referred to a specific type of ANN or identified which one was best for DHM problems. Different types have different advantages and disadvantages, so choosing which one to use for a particular problem depends on the problem and the proper network for solving it; it is more an art than a science.

In addition to determining which type of ANN is best for DHM problems, this thesis aimed to perform the training process, which is the core of using ANN, more generically and intelligently. Better network training yields improved results and more accurate prediction. To that end, this thesis examined the use of the general regression neural network (GRNN) to predict various DHM problems. GRNN's advantages include the following:

1. GRNN works quickly and without memory or training time problems. Some types of ANN that were previously used as common networks in DHM problems, like the feed forward neural network, experience problems in training when they encounter a large number of inputs and outputs and/or training cases.
2. GRNN smoothes out the regression curve between the training grids, so it can predict the system behavior using a small number of training cases. Thus, there is no need for a large number of training cases from expensive training sources like motion capture to train the network well and yield good general prediction for the system under various conditions.

3. GRNN training never converges to a poor solution because there is no iterative optimization in the training process. Hence, accepted results from this network are guarantees for any inputs within the training grid.
4. GRNN does not need many heuristic network parameters to be determined in advance for optimal network design. The Gaussian width (GW) is the only parameter that must be found for the network to perform well; this makes it easier for the user to construct a network with best performance. Other types of ANN require the user to specify the number of hidden layers, the neurons in each layer, the transfer function for each layer, and other parameters.

This thesis made the following contributions to the DHM field:

1. Introduced selection and use of GRNN to solve some DHM issues. The GRNN was selected because of its advantages that work best to handle the DHM problems. Then, GRNN was successfully used for three applications in different DHM fields, as described in Chapters 4, 5, and 6.
2. Developed a new strategy for determining the network parameters that improve the accuracy of predicting any system. Moreover, training and testing processes for any task were automated for GRNN, guaranteeing maximum performance. Then, the strategy was evaluated successfully on different DHM problems in Chapters 4 and 5. This strategy was also automated so that anyone could easily use the GRNN for best performance in any application.
3. Developed task-based human motion and posture predictions using GRNN. These broad applications allowed investigating the strengths and limitations in using GRNN in DHM applications. The strengths were studied in terms of predicting relatively large number of outputs from two different types, the speed of predicting this large number of outputs, and the achieved accuracy when predicting various tasks.

4. Developed algorithm to automate collecting the training cases for any motion task. The algorithm saved the consumed time in collecting training cases by changing all required input files each time before running the PD, and saving the results (new training case) consecutively under a new name.
5. Improved the speed of calculating human motion prediction by calculating the outputs from any inputs in a fraction of second. Filtered out system solutions so that all network predictions are considered accepted outputs (i.e., no bad or infeasible solutions are expected). This essentially introduces coupling ANN with PD to provide a faster predictive system.
6. Investigated potential issues when using GRNN to simulate posture prediction tasks that involve contact constraints or other conditions involving Cartesian locations. This thesis found serious limitation for using GRNN to predict the exact joint angles in touching point task. The thesis also provided an initial investigation as a platform for further study of ANN with posture prediction (i.e., evaluation of performance-measures (PMs) combinations).
7. Proposed an intelligent decision engine for human PMs. The engine was tested to choose the proper PM combinations for the posture prediction tasks presented Chapter 6. In addition, a new training strategy for the engine was proposed for indirectly extracting more reliable training cases from motion capture data. This application was new for ANN in DHM, where GRNN was used to predict the weights of PMs that are minimized in a posture prediction problem. The goal of this application was to develop a decision engine for selecting these PMs in a task-based manner and, eventually, to gain a thorough understanding of the relationships between these PMs for various tasks. In addition, this thesis proposed a mathematical formulation for indirectly extracting human PM weights from motion-captured postures. The proposed method was proved in Chapter 6.

7.2. Discussion

ANN as powerful tool for system prediction was applied successfully in this thesis. The hypothesis of successful improving predictive DHMs using ANN in this thesis was valid and promising. In general, GRNN was trained quickly and without experiencing any memory problems, no matter the size of the problem, in all applications. Moreover, the network predicted outputs from all types and numbers in a fraction of second.

The semi-automated process for training and testing the network and determining the network parameter (GW) that was introduced in Chapter 3 facilitated the use of GRNN without the need for understanding and setting its properties for each intended use. As such, many more applications and tasks could be predicted and applied by any user with this type of ANN. Moreover, the new heuristic strategy for determining the GW parameter worked well for all applications; the network outputs were highly accepted using the automatically chosen GW values. Generally speaking, it was found that system prediction depends on: 1) the input combination (i.e., how close to the point the training case is), 2) the number of training cases, and 3) the network properties, which were automatically selected in this study to have the maximum accuracy for such a task. Moreover, the type of inputs and outputs that form a task could be studied to improve the performance of the network used.

Generally, the results obtained from applying GRNN with the new training and testing strategy were comparable with those from the training sources (i.e., the exact results). The thesis successfully used the GRNN to predict various task-based motions and postures; it was generally able to produce very accurate results for both on- and off-grid points. Thus, selecting GRNN type of ANNs in this thesis to predict DHM applications was the right choice, because the results were generally accepted for different applications. Another challenge in this thesis was checking GRNN's ability to predict hundreds of outputs; this is a relatively large number and one of the limitations

when using other types of ANN to study human performance. In addition, the network was able to predict a large number of outputs of different types. In the motion prediction tasks in Chapter 4, the walking forward task combined joint angle profiles with joint torques at the same network outputs, while the jumping up on a box task had joint angle profiles with ground reaction forces (GRFs).

All network training and testing processes in this thesis was performed in MATLAB, which provides a solid ANN toolbox. This toolbox has built-in functions to construct, train, and save the network. Then, the saved network is packaged as standalone application that could be used as function in any visual studio programs like C++ and C#. Santos software is built mainly using C++ and C# codes, which allows calling the MATLAB functions directly from Santos software without opening MATLAB or having it installed on the used machine. By doing the above steps, the saved network could be used directly in Santos environment.

In the motion prediction tasks that were presented in Chapter 4, the training source, which was a PD algorithm, took hours to run the 52 training cases. The results produced from GRNN in a fraction of a second for both off- and on-grid test cases were subjectively and objectively comparable to those obtained from PD. Thus, the speed of GRNN prediction provides more options in terms of input ranges for the user to change the task input parameters and see immediate outputs. On the other hand, the obtained accurate predicted results from the network were related to the deterministic nature of predictive dynamics.

The studied motion tasks were chosen to examine GRNN's ability to address different issues. The walking task had 12 inputs and a relatively large number of two different types of outputs. The network was able to accurately predict all outputs. Jumping up on a box, which is complicated task with many constraints, needed high accuracy in predicting its outputs, because the hands and feet should be in contact with the box at some points of the task time. The constructed network for this task addressed

all these issues successfully with outstanding fast and accurate results. The results from this chapter prove the assumption of using ANN to predict human motions realistically and quickly. Thus, there is a great potential for GRNN to be widely used in real-time, task-based motion prediction and to perform like a human brain, developing and training continuously.

The GRNN type of ANN has never been used in posture prediction problems. In Chapter 5, two different task-based posture predictions were performed for touching a point on the front side of the body with and without external load. A relatively small number of training cases was used to train the network to predict statistically accepted results. On the other hand, the visual results showed that there is some accuracy limitation for touching the exact points. Thus, the contact problem (touching the exact point or location) in this chapter was serious. Santos failed in exactly touching the target point for most of the tested cases, even the on-grid cases. To solve the contact problem in posture prediction tasks, two options could be applied: first, collecting more training cases for better network training and more accurate prediction results, and second, adding constraints to the network construction to force the predicted postures from the network to be exactly in the proper position**.**

The decision engine for human PMs that was introduced in Chapter 6 provided the initial step in generalizing the PM selections at various posture and motion prediction tasks under different conditions. The results of this engine showed that the GRNN predicted the PM weights well for both tasks. Predicting different on- and off-grid test cases with visually accepted results, subjective validation, indicated that this method could be helpful, because it saves the time of trying many PM weight collections until the postures look better. From applying the engine in both tasks, general trends were drawn: people tend to minimize their joint torques in cases where there is applied load, and to minimize the movement (joint displacement) when no load exists.

In Chapter 6, the new method that used motion-captured postures as a training source succeeded in extracting the PM weights from given Mo-cap joint angles for one training case. This method is the first approach to obtain information about PM from a reliable source like Mo-cap. The extracted PM weights from the case using this method showed that the four PMs are important in order to predict realistic human posture. The optimization solution provided values for the four weights, which means that all PMs contributed in producing the actual posture. This training method could be populated to be used over many postures at various tasks. Given enough Mo-cap training data for a task, this method will be able to train GRNN properly to predict that task. Further proper prediction for PM weights at different tasks under many conditions could lead to a thorough understanding of the relationships between the PMs, which would ultimately lead to understanding how and why people think as they do when accomplishing a task.

7.3. Future Work

The successful use of GRNN in solving DHM problems that presented in this thesis could be expanded to study more human performance applications including, but not limited to, grasping, obstacle avoidance, and human-workplace design optimization. Given the application of ANN to motion prediction, posture prediction, and decision engine for human PMs, the use of ANN as decision engine is the most lacking in DHMs. And using ANN in decision engines, in specific, will have the most impact on the field of DHM, because understanding and predicting human drivers (PMs) allows more robust motion and posture predictions.

Some limitations need to be addressed for the general use of GRNN in DHM field, including: 1) automatic task definition, 2) the optimal number of training cases for a task, and 3) extrapolation of prediction capability for off-grid points, which needs studying other types of ANN. Future work would include the following:

1. Investigate functions other than the Gaussian function that is used as radial transfer function in GRNN.
2. Normalize the network outputs in case of having different types with different ranges to have one range for them all at the same network. Similar to the network inputs, normalizing the outputs could improve the network prediction ability.
3. Reconstruct GRNN with different GW values at different neurons in the hidden layer. Each neuron will have its own GW to maximize the predicted outputs at the level of the neurons instead of the general network.

Along with the successful applications of GRNN in predicting human motion in Chapter 4, tasks that include other factors and more complicated motions need to be studied in the future. To solve the problem of accuracy differences among different network outputs, specific joints could be targeted for further analysis when performing motion tasks. Studying joint significance is performed by studying the network neurons that contributes in the resulting motion profile.

Regarding the limitations when applying GRNN to predict the task of touching point in the front side of the body, the network is unable to predict the proper set of joint angles to touch the target point exactly. In addition, to generalize the prediction capability for GRNN in posture prediction, the network should be used for further posture prediction studies. Generally, to solve the contact problem (accuracy of touching the target point) in posture prediction tasks, there are three potential options:

1. Collecting many training cases to decrease the gaps in the training grid. Then, the new methodology for determining best GW will be able to set small GW and predict the task more accurately.
2. Adding constraints to the network construction to force the predicted postures from the network to be exactly in the proper position.
3. Training the network to predict joint center locations instead of joint angles may succeed in producing more accurate results in posture prediction tasks. Predicting

joint angles with small errors produces highly inaccurate results, because any error in predicting joint angles produces severe problems in Cartesian space results. On the other hand, predicting joint centers with small error should still provide accurate results.

The new training method presented in Chapter 6 for using Mo-cap data to extract the PM weights is unique and promising. This method could provide the GRNN with reliable training data to predict the proper PM weights for different human PMs in a task-based manner. Eventually, GRNN could perform as a decision engine for human PMs that control posture prediction and many other DHM fields. Thus, future work should entail obtaining the PM weights from additional Mo-cap data using the proposed method to train the network. The potential long-term work from this thesis is proposed in the following points:

1. Improving the accuracy and task-based performance for the network, which requires developing in-house ANN (i.e., building our own network). This might be done using some or all of the following: a) separate networks for multi-task predictions, b) expand the network to incorporate constraints, and c) investigate extrapolating the predicted task, which could be performed using types of ANN other than the GRNN.
2. Generalizing the task definition to have ANN work for various applications. This generalization could lead to predict a set of tasks (scenario) successfully.
3. Studying what drives human performance in terms of PMs and neurons. Automating the PMs selection based-on the task and its conditions will help understanding which PMs drive human decisions at various tasks. In addition, connecting the network's neurons to those in the natural brain by studying each neuron contribution in the final outputs, this will refer to the importance of that specific neuron in the task.

4. Using the network training process to study the difference between well trained and poorly trained individuals. This could be performed by training the network to predict specific task using larger and/or fewer number of training cases, which reflects the individual's experience in performing that task.

APPENDIX A

HUMAN MODEL

Currently, there are many human modeling software programs that analyze and study human behaviors realistically. To study what drives human performance, however, we need a well-developed human model with high fidelity. A good human model has a sufficient number of DOFs, real joint range of motion (ROM), and real anthropometry. In this thesis, we will use the Santos human model, which is one of the most advanced human models that have been developed to date. The Santos virtual human software was built by the Virtual Soldier Research (VSR) program at The University of Iowa (Figure A.1). This human model mimics the human body by having 55 DOFs and all links that connect these DOFs (Abdel-Malek et al., 2007). Santos's skeleton is modeled to move as a series of links where each pair of links is connected by one or more revolute joints. Each joint could contribute in one or more DOFs and, depending on the task, some or all joint revolutions will produce the motion of the skeleton as one composite.

In Figure A.1, Santos's joints are located where different links meet. At the joint location, there are one or more DOFs depending on the anatomical structure of that joint in the natural human. The green cylinders represent the different DOFs provided by various joints, and black arrows (L1 to L28) show the links. In addition, each DOF has ROM equal to that of an average human. Hence, Santos shows by validation real and trustable results for many applications that are done by the VSR group. Therefore, we will depend on this human model for all parts of this thesis.

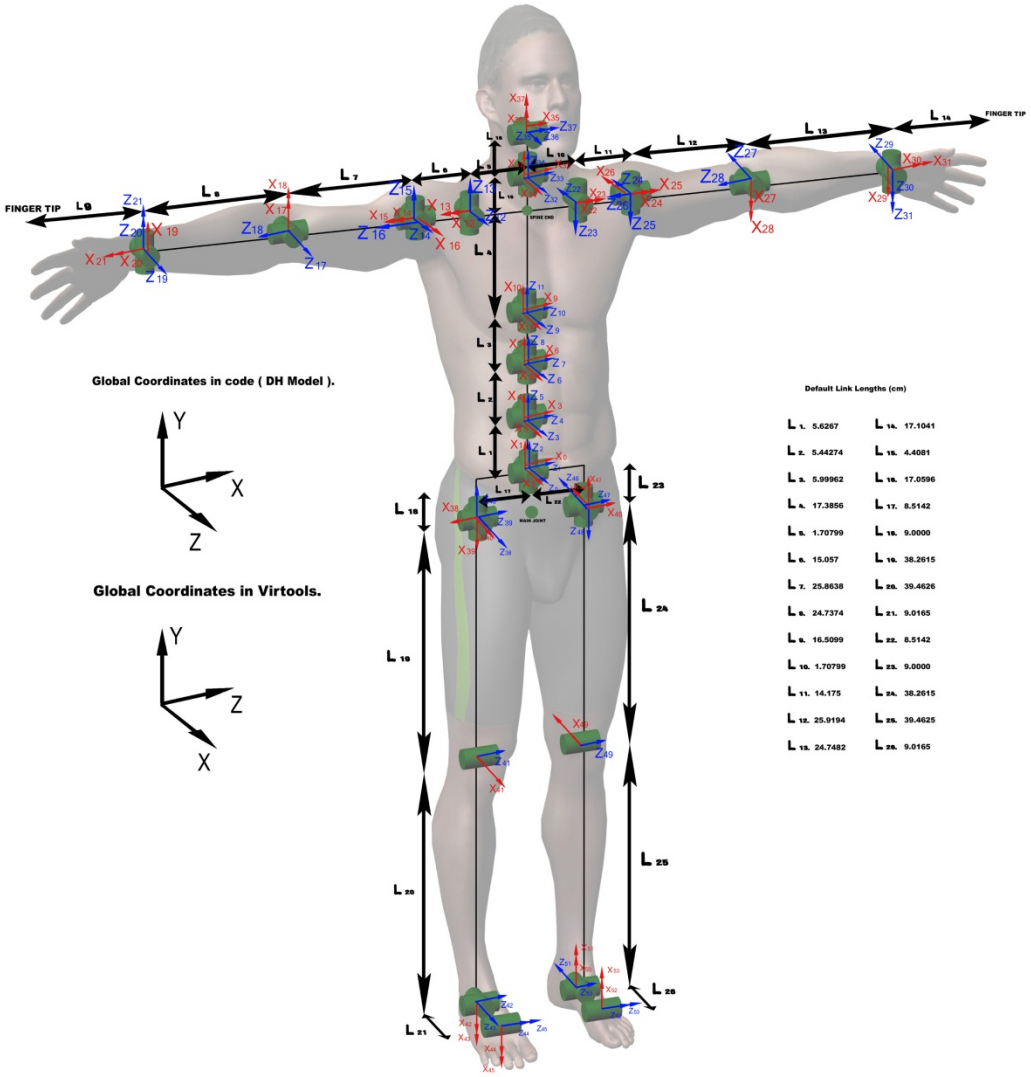

Figure A.1: The Virtual Human Santos.

APPENDIX B

TABLES OF TRAINING CASES FOR ALL TASKS

Table B.1: Values of the input parameters for the training cases in walking forward task.

Case #	Velocity	BP weight	Link 1	Link 2	Link 3	Link 4	ROM 1 Lower	ROM 1 upper	ROM 2 Lower	ROM 2 Upper	ROM 3 Lower	ROM 3 Upper
1	0.8	0	0.078	0.434	0.395	0.121	-123.3	8.7	5	149.7	7.3	71.6
2	0.8	-157.5	0.078	0.434	0.395	0.121	-123.3	8.7	5	149.7	7.3	71.6
3	0.8	-315	0.078	0.434	0.395	0.121	-123.3	8.7	5	149.7	7.3	71.6
4	1.6	0	0.078	0.434	0.395	0.121	-123.3	8.7	5	149.7	7.3	71.6
5	1.6	-157.5	0.078	0.434	0.395	0.121	-123.3	8.7	5	149.7	7.3	71.6
6	0.8	0	0.088	0.445	0.424	0.117	-123.3	8.7	5	149.7	7.3	71.6
7	0.8	-157.5	0.088	0.445	0.424	0.117	-123.3	8.7	5	149.7	7.3	71.6
8	0.8	-315	0.088	0.445	0.424	0.117	-123.3	8.7	5	149.7	7.3	71.6
9	1.6	-157.5	0.088	0.445	0.424	0.117	-123.3	8.7	5	149.7	7.3	71.6
10	1.6	-315	0.088	0.445	0.424	0.117	-123.3	8.7	5	149.7	7.3	71.6
11	0.8	0	0.098	0.456	0.454	0.113	-123.3	8.7	5	149.7	7.3	71.6
12	0.8	-157.5	0.098	0.456	0.454	0.113	-123.3	8.7	5	149.7	7.3	71.6
13	0.8	-315	0.098	0.456	0.454	0.113	-123.3	8.7	5	149.7	7.3	71.6
14	1.6	0	0.098	0.456	0.454	0.113	-123.3	8.7	5	149.7	7.3	71.6
15	1.6	-157.5	0.098	0.456	0.454	0.113	-123.3	8.7	5	149.7	7.3	71.6
16	1.6	-315	0.098	0.456	0.454	0.113	-123.3	8.7	5	149.7	7.3	71.6
17	0.8	0	0.088	0.445	0.424	0.117	-105	5	5	149.7	7.3	71.6
18	0.8	-157.5	0.088	0.445	0.424	0.117	-105	5	5	149.7	7.3	71.6
19	0.8	-315	0.088	0.445	0.424	0.117	-105	5	5	149.7	7.3	71.6
20	1.6	0	0.088	0.445	0.424	0.117	-105	5	5	149.7	7.3	71.6
21	1.6	-157.5	0.088	0.445	0.424	0.117	-105	5	5	149.7	7.3	71.6
22	1.6	-315	0.088	0.445	0.424	0.117	-105	5	5	149.7	7.3	71.6
23	0.8	0	0.088	0.445	0.424	0.117	-90	2	5	149.7	7.3	71.6
24	0.8	-157.5	0.088	0.445	0.424	0.117	-90	2	5	149.7	7.3	71.6
25	0.8	-315	0.088	0.445	0.424	0.117	-90	2	5	149.7	7.3	71.6
26	1.6	0	0.088	0.445	0.424	0.117	-90	2	5	149.7	7.3	71.6
27	1.6	-157.5	0.088	0.445	0.424	0.117	-90	2	5	149.7	7.3	71.6

Table B.1: Continued.

28	1.6	-315	0.088	0.445	0.424	0.117	-90	2	5	149.7	7.3	71.6
29	0.8	0	0.088	0.445	0.424	0.117	-123.3	8.7	10	130	7.3	71.6
30	0.8	-157.5	0.088	0.445	0.424	0.117	-123.3	8.7	10	130	7.3	71.6
31	0.8	-315	0.088	0.445	0.424	0.117	-123.3	8.7	10	130	7.3	71.6
32	1.6	0	0.088	0.445	0.424	0.117	-123.3	8.7	10	130	7.3	71.6
33	1.6	-157.5	0.088	0.445	0.424	0.117	-123.3	8.7	10	130	7.3	71.6
34	1.6	-315	0.088	0.445	0.424	0.117	-123.3	8.7	10	130	7.3	71.6
35	0.8	0	0.088	0.445	0.424	0.117	-123.3	8.7	20	110	7.3	71.6
36	0.8	-157.5	0.088	0.445	0.424	0.117	-123.3	8.7	20	110	7.3	71.6
37	0.8	-315	0.088	0.445	0.424	0.117	-123.3	8.7	20	110	7.3	71.6
38	1.6	0	0.088	0.445	0.424	0.117	-123.3	8.7	20	110	7.3	71.6
39	1.6	-157.5	0.088	0.445	0.424	0.117	-123.3	8.7	20	110	7.3	71.6
40	1.6	-315	0.088	0.445	0.424	0.117	-123.3	8.7	20	110	7.3	71.6
41	0.8	0	0.088	0.445	0.424	0.117	-123.3	8.7	5	149.7	15	60
42	0.8	-157.5	0.088	0.445	0.424	0.117	-123.3	8.7	5	149.7	15	60
43	0.8	-315	0.088	0.445	0.424	0.117	-123.3	8.7	5	149.7	15	60
44	1.6	0	0.088	0.445	0.424	0.117	-123.3	8.7	5	149.7	15	60
45	1.6	-157.5	0.088	0.445	0.424	0.117	-123.3	8.7	5	149.7	15	60
46	1.6	-315	0.088	0.445	0.424	0.117	-123.3	8.7	5	149.7	15	60
47	0.8	0	0.088	0.445	0.424	0.117	-123.3	8.7	5	149.7	20	50
48	0.8	-157.5	0.088	0.445	0.424	0.117	-123.3	8.7	5	149.7	20	50
49	0.8	-315	0.088	0.445	0.424	0.117	-123.3	8.7	5	149.7	20	50
50	1.6	0	0.088	0.445	0.424	0.117	-123.3	8.7	5	149.7	20	50
51	1.6	-157.5	0.088	0.445	0.424	0.117	-123.3	8.7	5	149.7	20	50
52	1.6	-315	0.088	0.445	0.424	0.117	-123.3	8.7	5	149.7	20	50

Table B.2: Values of the input parameters for the training cases in jumping up on box task.

case #	Box height	Link 1	Link 2	Link 3	Link 4
1	50	9	38	39	9
2	55	9	38	39	9
3	60	9	38	39	9
4	65	9	38	39	9
5	70	9	38	39	9
6	75	9	38	39	9
7	80	9	38	39	9
8	85	9	38	39	9
9	90	9	38	39	9
10	95	9	38	39	9
11	100	9	38	39	9
12	50	7.8	43	39	12
13	55	7.8	43	39	12
14	60	7.8	43	39	12
15	65	7.8	43	39	12
16	70	7.8	43	39	12
17	75	7.8	43	39	12
18	80	7.8	43	39	12
19	85	7.8	43	39	12
20	90	7.8	43	39	12
21	95	7.8	43	39	12
22	100	7.8	43	39	12

Table B.3: Values of the input parameters for the training cases in the task of posture prediction without external load.

Case #	X	Y	Z	R. shoulder 1 Lower	R. shoulder 1 Upper	R. shoulder 2 Lower	R. shoulder 2 Upper	R. Elbow Lower	R. Elbow Upper
1	6	-51	-67	-23	123.5	-19	111	-148.5	-12.5
2	-40	-40	-73	-23	123.5	-19	111	-148.5	-12.5
3	-64	51	-77	-23	123.5	-19	111	-148.5	-12.5
4	-20	39	-98	-23	123.5	-19	111	-148.5	-12.5
5	29	44	-96	-23	123.5	-19	111	-148.5	-12.5
6	25	106	-40	-23	123.5	-19	111	-148.5	-12.5
7	-6	110	-32	-23	123.5	-19	111	-148.5	-12.5
8	-39	99	-40	-23	123.5	-19	111	-148.5	-12.5
9	-49	100	-22	-23	123.5	-19	111	-148.5	-12.5
10	-20	105	-26	-23	123.5	-19	111	-148.5	-12.5
11	28	106	-28	-23	123.5	-19	111	-148.5	-12.5
12	70	15	-44	-23	123.5	-19	111	-148.5	-12.5
13	-2	-3	-26	-23	123.5	-19	111	-148.5	-12.5
14	-93	23	-41	-23	123.5	-19	111	-148.5	-12.5
15	-50	-37	-55	-23	123.5	-19	111	-148.5	-12.5
16	37	-36	-50	-23	123.5	-19	111	-148.5	-12.5
17	-3	-1	-95	-23	123.5	-5	50	-148.5	-12.5
18	34	77	-30	-23	123.5	-5	50	-148.5	-12.5
19	-31	67	-42	-23	123.5	-5	50	-148.5	-12.5
20	37	120	3	-23	123.5	-5	50	-148.5	-12.5
21	-76	20	-61	-23	123.5	-5	50	-148.5	-12.5
22	-3	108	-2	-5	60	-19	111	-148.5	-12.5
23	-1	56	-37	-5	60	-19	111	-148.5	-12.5
24	28	-17	-67	-5	60	-19	111	-148.5	-12.5
25	61	-38	-17	-5	60	-19	111	-148.5	-12.5
26	-102	49	-22	-5	60	-19	111	-148.5	-12.5
27	-43	-56	-37	-23	123.5	-19	111	-80	-40
28	-12	95	-57	-23	123.5	-19	111	-80	-40
29	55	71	-60	-23	123.5	-19	111	-80	-40
30	-49	-55	-33	-23	123.5	-19	111	-80	-40
31	77	74	-21	-23	123.5	-19	111	-80	-40

Table B.4: Values of the input parameters for the training cases in the task of posture prediction with external load.

Case #	X	Y	Z	R. shoulder 1 Lower	R. shoulder 1 Upper	R. shoulder 2 Lower	R. shoulder 2 Upper	R. Elbow Lower	R. Elbow Upper
1	-76	-20	-33	-23	123.5	-19	111	-148.5	-12.5
2	-16	-16	-35	-23	123.5	-19	111	-148.5	-12.5
3	17	-11	-34	-23	123.5	-19	111	-148.5	-12.5
4	33	-22	-30	-23	123.5	-19	111	-148.5	-12.5
5	32	-41	-44	-23	123.5	-19	111	-148.5	-12.5
6	-21	-17	-61	-23	123.5	-19	111	-148.5	-12.5
7	-49	-47	-41	-23	123.5	-19	111	-148.5	-12.5
8	-60	34	-31	-23	123.5	-19	111	-148.5	-12.5
9	-5	38	-19	-23	123.5	-19	111	-148.5	-12.5
10	59	42	-35	-23	123.5	-19	111	-148.5	-12.5
11	11	97	-18	-23	123.5	-19	111	-148.5	-12.5
12	-19	102	-14	-23	123.5	-19	111	-148.5	-12.5
13	-15	107	-13	-23	123.5	-19	111	-148.5	-12.5
14	-24	103	-21	-23	123.5	-19	111	-148.5	-12.5
15	-10	85	-50	-23	123.5	-19	111	-148.5	-12.5
16	19	103	-24	-23	123.5	-19	111	-148.5	-12.5
17	23	29	-76	-23	123.5	-19	111	-148.5	-12.5
18	-3	14	-73	-23	123.5	-19	111	-148.5	-12.5
19	-50	62	-54	-23	123.5	-19	111	-148.5	-12.5
20	12	19	-72	-23	123.5	-19	111	-148.5	-12.5
21	17	-6	-82	-23	123.5	-19	111	-148.5	-12.5
22	-51	14	-82	-23	123.5	-19	111	-148.5	-12.5
23	-43	11	-65	-23	123.5	0	50	-148.5	-12.5
24	10	-19	-60	-23	123.5	0	50	-148.5	-12.5
25	18	70	-33	-23	123.5	0	50	-148.5	-12.5
26	-17	71	-35	-23	123.5	0	50	-148.5	-12.5
27	-41	70	-39	-5	50	-19	111	-148.5	-12.5
28	38	68	-40	-5	50	-19	111	-148.5	-12.5
29	24	-30	-47	-5	50	-19	111	-148.5	-12.5
30	-49	-30	-49	-5	50	-19	111	-148.5	-12.5
31	-40	17	-65	-23	123.5	-19	111	-70	-30
32	12	8	-68	-23	123.5	-19	111	-70	-30
33	7	95	-36	-23	123.5	-19	111	-70	-30
34	-21	107	-23	-23	123.5	-19	111	-70	-30

Table B.5: Values of the input parameters for the training cases in extracting the performance measures (PMs) for the task of posture prediction without external load.

Case #	X	Y	Z	R. shoulder 1 Lower	R. shoulder 1 Upper	R. shoulder 2 Lower	R. shoulder 2 Upper	R. Elbow Lower	R. Elbow Upper
1	-63	45	0	-23	123.5	-19	111	-148.5	-12.5
2	-66	69	-54	-23	123.5	-19	111	-148.5	-12.5
3	0	33	-23	-23	123.5	-19	111	-148.5	-12.5
4	-7	85	-70	-23	123.5	-19	111	-148.5	-12.5
5	-12	37	-102	-23	123.5	-19	111	-148.5	-12.5
6	51	35	-15	-23	123.5	-19	111	-148.5	-12.5
7	37	7	-88	-23	123.5	-19	111	-148.5	-12.5
8	46	46	-18	-23	123.5	-19	111	-148.5	-12.5
9	-23	-50	-58	-23	123.5	-19	111	-148.5	-12.5
10	52	99	1	-23	123.5	-19	111	-148.5	-12.5
11	52	99	-26	-23	123.5	-19	111	-148.5	-12.5
12	0	91	-18	-23	123.5	-19	111	-148.5	-12.5
13	-46	103	-17	-23	123.5	-19	111	-148.5	-12.5
14	-97	70	-10	-23	123.5	-19	111	-148.5	-12.5
15	-56	-2	-89	-5	60	-5	50	-80	-40
16	-21	69	-82	-5	60	-5	50	-80	-40
17	5	38	-57	-5	60	-5	50	-80	-40
18	47	85	-56	-5	60	-5	50	-80	-40
19	-3	21	-52	-10	110	-19	59	-120	-20
20	40	-13	-57	-8	90	-10	80	-130	-20
21	-37	94	-37	-17	70	-15	100	-90	-35

Table B.6: Values of the input parameters for the training cases in extracting the performance measures (PMs) for the task of posture prediction with external load.

Case #	X	Y	Z	R. shoulder 1 Lower	R. shoulder 1 Upper	R. shoulder 2 Lower	R. shoulder 2 Upper	R. Elbow Lower	R. Elbow Upper
1	-63	45	0	-23	123.5	-19	111	-148.5	-12.5
2	-66	69	-54	-23	123.5	-19	111	-148.5	-12.5
3	0	33	-23	-23	123.5	-19	111	-148.5	-12.5
4	-7	85	-70	-23	123.5	-19	111	-148.5	-12.5
5	-12	37	-102	-23	123.5	-19	111	-148.5	-12.5
6	51	35	-15	-23	123.5	-19	111	-148.5	-12.5
7	37	7	-88	-23	123.5	-19	111	-148.5	-12.5
8	46	46	-18	-23	123.5	-19	111	-148.5	-12.5
9	-23	-50	-58	-23	123.5	-19	111	-148.5	-12.5
10	52	99	1	-23	123.5	-19	111	-148.5	-12.5
11	52	99	-26	-23	123.5	-19	111	-148.5	-12.5
12	0	91	-18	-23	123.5	-19	111	-148.5	-12.5
13	-46	103	-17	-23	123.5	-19	111	-148.5	-12.5
14	-97	70	-10	-23	123.5	-19	111	-148.5	-12.5
15	-56	-2	-89	-5	60	-5	50	-80	-40
16	-21	69	-82	-5	60	-5	50	-80	-40
17	5	38	-57	-5	60	-5	50	-80	-40
18	47	85	-56	-5	60	-5	50	-80	-40
19	-3	21	-52	-10	110	-19	59	-120	-20
20	40	-13	-57	-8	90	-10	80	-130	-20
21	-37	94	-37	-17	70	-15	100	-90	-35

REFERENCES

Abdel-Malek, K., W. Yu, M. Jaber, and J. Duncan. 2001. “Realistic posture prediction for maximum dexterity.” SAE Digital Human Modeling Conference, Arlington, VA.

Abdel-Malek, K., J. Yang, J. Kim, T. Marler, S. Beck, C. Swan, et al. 2007. “Development of the virtual-human Santos™.” *Digital Human Modeling* 4561: 490–499.

Buhmann, M.D. 2003. *Radial Basis Functions: Theory and Implementations*. Cambridge: Cambridge University Press.

Bu, N., M. Okamoto, and T. Tsuji. 2009. “A hybrid motion classification approach for EMG-based human robot interfaces using Bayesian and neural networks.” *IEEE Trans. Robot.* 25(3): 502–511.

Bishop, C.M. 1995. *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press.

Beck, D.J., and D.B. Chan. 1992. “An evaluation of inverse kinematics models for posture prediction.” In *Computer Applications in Ergonomics, Occupational Safety and Health*, edited by M. Mattila and W. Karkowski, 329–336. Amsterdam: Elsevier.

Chaffin, D.B. 2001. “On simulating human reach motions for ergonomics analyses.” Paper presented at Computer-Aided Ergonomics and Safety Conference, Maui, HI.

Collobert, R., and J. Weston. 2008. “A unified architecture for natural language processing: Deep neural networks with multitask learning.” In *Proceedings of the 25th International Conference on Machine Learning (ICML-08)*, 160–167.

Coit, D.W., B.T. Jackson, and A.E. Smith. 1998. “Static neural network process models: considerations and case studies.” *International Journal of Production Research* 36(11): 2953–2967.

Gill, P., W. Murray, and A. Saunders. 2002. “SNOPT: An SQP algorithm for large-scale constrained optimization.” *SIAM Journal of Optimization* 12(4): 979–1006.

Hahn, M.E., A.M. Farley, V. Lin, and L.S. Chou. 2005. “Neural network estimation of balance control during locomotion.” *Journal of Biomechanics* 38: 717–23.

Jaremko, J.L., P. Poncet, J. Ronsky, J. Harder, J. Dansereau, H. Labelle, and R.F. Zernicke. 2002. “Genetic algorithm-neural network estimation of Cobb angle from torso asymmetry in scoliosis.” *J. Biomech. Eng*. 5: 496–503.

Jung, E.S., and S.J. Park. 1994. “Prediction of human reach posture using a neural network for ergonomic man models.” *Computer and Industrial Engineering* 27(1-4): 369–372.

Kang, B., B. Kim, S. Park, and H. Kim. 2007. “Modeling of artificial neural network for the prediction of the multi-joint stiffness in dynamic condition.” In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.,* 1840–1845.

Liu, Q., T. Marler, J. Yang, J. Kim, and C. Harrison. 2009. “Posture prediction with external loads – a pilot study.” Paper presented at the SAE 2009 World Congress, Detroit, MI.

Liegeois, A. 1977. “Automatic supervisory control of the configuration and behavior of multibody mechanisms.” *IEEE Transactions on Systems. Man, and Cybernetics* 7(12): 868–871.

Li, Y., C. Li, and R. Song. 2008. “A new hybrid algorithm of dynamic obstacle avoidance based on dynamic rolling planning and RBFNN.” In *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, 2064–2068.

Lawrence, J., and J. Fredrickson. 1998. *BrainMaker User’s Guide and Reference Manual*, 7th ed. Nevada City, CA: California Scientific Software.

Marler, R.T., and J.S. Arora. 2004. “Survey of multi-objective optimization methods for engineering.” *Structural and Multidisciplinary Optimization* 26(6): 369–395.

Marler, R.T., J. Yang, J.S. Arora, and K. Abdel-Malek. 2005a. “Study of bi-criterion upper body posture prediction using pareto optimal sets.” Paper presented at the IASTED International Conference on Modeling, Simulation, and Optimization, Oranjestad, Aruba.

Marler, R.T., S. Rahmatalla, M. Shanahan, and K. Abdel-Malek. 2005b. “A new discomfort function for optimization-based posture prediction.” Paper presented at the SAE Human Modeling for Design and Engineering Conference, Iowa City, IA.

Marler, R.T., and J.S. Arora. 2010. “The weighted sum method for multi-objective optimization: new insights.” *Structural & Multidisciplinary Optimization* 41: 853–862.

Marler, T., L. Knake, and R. Johnson R. 2011. “Optimization-based posture prediction for analysis of box lifting tasks.” *Digital Human Modeling*, *Lecture Notes in Computer Science*, 6777(2011): 151–160.

Mi, Z. 2004. “Task-based prediction of upper body motion,” PhD diss., University of Iowa.

Ma, L., W. Zhang, D. Chablat, F. Bennis, and F. Guillaume. 2009. “Multi-objective optimization method for posture prediction and analysis with consideration of fatigue effect and its application case.” *Computers & Industrial Engineering* 57(4): 1235–1246.

Moody, J., and C.J. Darken. 1989. “Fast learning in units of locally-tuned processing units.” *Neural Comp*. 1(2): 281–294.

Nishide, S., T. Ogata, J. Tani, K. Komatani, and H.G. Okuno. 2009. “Analysis of motion searching based on reliable predictability using recurrent neural network.” Paper presented at the 2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Singapore.

Najmaei, N., and M. Kermani. 2010. “Prediction-based reactive control strategy for human-robot interactions.” In Proceedings of *IEEE Int. Conf. Robot. Autom.*, 3434–3439.

Perez, M.A., and M.A. Nussbaum. 2008. “A neural network model for predicting postures during non-repetitive manual materials handling tasks.” *Ergonomics* 51(10): 1549–1564.

Park, S.I., H.J. Shin, and S.Y. Shin. 2002. “On-line locomotion generation based on motion blending.” In *Proceedings of the ACM SIGGRAPH Symposium on Computer Animation*, 105–112.

Rezzoug, N., and P. Gorce. 2008. “Prediction of fingers posture using artificial neural networks.” *Journal of Biomechanics* 41(12): 2743–2749.

Riffard, V., and P. Chedmail. 1996. “Optimal posture of a human operator and CAD in robotics.” In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1199–1204.

Specht, D. F. 1991. “A general regression neural network.” *IEEE Trans. Neural Network* 2(6): 568–576.

Tani, J, R. Nishimoto, J. Namikawa, and M. Ito. 2008. “Codevelopmental learning between human and humanoid robot using a dynamic neural-network model.” *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics* 38: 43–59.

Twomey, J.-M., and A.E. Smith. 1998. “Bias and variance of validation methods for function approximation neural networks under conditions of sparse data.” *IEEE Transactions on Systems, Man and Cybernetics* 28(3): 417–430.

Vogl, T.P., J.K. Mangis, A.K. Rigler, W.T. Zink, and D.L. Alkon. 1998. “Accelerating the convergence of the back propagation method.” *Biological Cybernetics* 59: 257–263.

Wasserman, P.D. 1993. *Advanced Methods in Neural Computing*. New York: Van Nostrand Reinhold.

Xiang, Y. 2008. “Optimization-based dynamic human walking prediction,” PhD diss., University of Iowa.

Xiang, Y., H.J. Chung, J. Kim, R. Bhatt, S. Rahmatalla, J. Yang, T. Marler, J. Arora, and K. Abdel-Malek. 2010. “Predictive dynamics: an optimization based novel approach for human motion simulation.” *Structural and Multidisciplinary Optimization* 41 (3): 465–479.

Yang, J., R.T. Marler, H. Kim, J. Arora, and K. Abdel-Malek. 2004. “Multi-objective optimization for upper body posture prediction.” Paper presented at the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY.

Yang, J., T. Marler, and S. Rahmatalla. 2011. “Multi-objective optimization-based kinematic posture prediction: development and validation.” *Robotica* 29: 245–253.

Yu, W. 2001. “Optimal placement of serial manipulators,” PhD diss., University of Iowa.

Zha, X. 2003. “Soft computing framework for intelligent human-machine system design, simulation and optimization.” *Soft Computing* 7(3): 184–98.

Zhao, H., G. Zheng, and W. Wen. 2010. “Human-machine posture prediction and working efficiency evaluation of virtual human using radial basis function neural network.” In *Proceedings of Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference*, 406–410.

Zhang, B., I. Horváth, J.F.M. Molenbroek, and C. Snijders. 2010. “Using artificial neural networks for human body posture prediction.” *International Journal of Industrial Ergonomics* 40(4): 414–424.