Neural Network for Large-Scale Problems, with Application to Human Motion

Mohammad Bataineh[1,*] and Timothy Marler[2]

[1]*Humana Inc.; Louisville, KY 40202, USA*

[2]*RAND Corporation; Santa Monica, CA 90401, USA*

Abstract: As the potential applications for artificial intelligence, and thus neural networks expand, and as the prevalence of big data increases, the need for improved training in neural networks that leverage data sets efficiently will soon surface. In addition, research in the field of human simulation has led to significant advancements in quality, time, and cost management for products like military and athletic equipment and vehicles. There is, however, a critical need for human simulation models to run in real time, especially those with large-scale problems like motion prediction (a single motion problem involves prediction of between 500-700 outputs). Hence, this work addresses both challenges by introducing a modified training process for an artificial neural network (ANN) that is capable of mitigating memory issues and improving accuracy with large scale problems that involve minimal taring data. The new modified ANN design is successfully tested on two common human-motion tasks, walking and going prone. Through comparison with a benchmark ANN, the results of the new network are shown to be accurate objectively and subjectively.

Keywords: Artificial neural network, Optimization, Digital human modeling, Motion prediction, Large-scale problems.

1. INTRODUCTION

With a growing dependence on, and availability of big data, methods for solving large-scale problems are becoming increasingly important with far reaching applications. Within this context, designing systems for machine learning that have limited training sets presents a particularly challenging problem for typical neural networks. Thus, this paper presents a new artificial neural network (ANN) that is specifically designed for such problems. This network is applied to computational problems for predicting dynamic human motion. To be sure, there are many methods for simulating tasks and scenarios as executed by a digital human model (DHM). One such method is called predictive dynamics (PD), which is used to predict dynamic human motion [1]. Predictive dynamics is a physics-based algorithm that consists of an optimization problem with hundreds of design variables and thousands of constraints.

Although PD simulations can be useful, they can also be computationally expensive, and more complex tasks can not necessarily be simulated in real time. Furthermore, even small changes to the task conditions, can affect the run time significantly. Thus, in cases where PD is used to train an ANN, the number of training cases can be limited due to the computational time and cost associated with collecting training data. The necessary manual post-processing procedures can also be prohibitive.

In addition to being computationally demanding, PD problems are often relatively large with respect to the number of outputs; hundreds of outputs (500-700) are required in order to predict a single problem (*i.e*., one motion task). Thus, there is a need for a computational model that can provide accurate simulations of dynamic human motion in real time, based on PD results.

In response to the special needs of large-scale DHM problems with limited training data available, this work proposes the use of a newly developed radial-basis network (RBN), called Opt_RBN, [2]. Although the Opt_RBN design is proven to provide accurate prediction results for applications with a reduced number of training cases, applying this design to large-scale applications requires additional enhancements. Opt_RBN experiences a memory issue during the optimization step. Therefore, this work introduces new modified algorithms for the Opt_RBN training process in order to address the memory issue without affecting the network performance. The resulting modified Opt_RBN provides real-time prediction of a PD motion problem that can be trained on any PC.

In general, the use of ANN and other machine learning techniques for large-scale problems focuses on problems that have a large number of training data [3-6] due to the recent advancement in online web services, software, and devices that provide massive

*Address correspondence to this author at the Humana Inc.; Louisville, KY 40202, USA; Tel: +1-319-333-2433; Fax: +1-502-580-1041;
E-mail: bataineh.moe@gmail.com; tmarler@rand.org

E-ISSN: 2410-2938/16

amounts of data for training purposes. That is, the term *large-scale* typically refers to the size of the training data set. There is limited use of ANNs in applications that have limited numbers of training data like those produced from high-fidelity finite element analysis models or complex optimization and image processing systems [7-11]. Although these applications have recently been expanded to make them faster with advanced techniques like parallel algorithms [12-14], they still experience running delays that make them difficult torunin real time without the assistance of predictive models like ANN. Thus, ANNs and other techniques have been investigated and widely used in the recent years to help address various complicated practical problems. Such advancement is notable primarily in large-scale image classifications and video analysis [5, 15-18] to replace the original computationally expensive or limited-accuracy models. In these works, the goal is to find the most appropriate approaches that increase the results' accuracy, given that there is always enough data to train and evaluate the models. Consequently, the remaining issue is the trade-off between the available training cases and the accuracy level of the resulting model [19]. However, it is still crucial to solve the same problems, and with comparable accuracy, but when limited training cases are available. There is minimal work that targets increased accuracy with limited available data.

With respect to the use of ANNs in applications related to motion prediction, recent advancements have been achieved in machines and humanoid interfaces [20-23]. Although various types of ANNs have been successfully used for predicting motion and movements, the humanoid and animations that are produced in these applications involve relatively simple models with a small number of degrees of freedom (DOF). Full-DOF human motion prediction is still an active area of research due to the complexity of designing and simulating the actual human models. The proposed work focuses on simulating the motions of the Santos DHM [24]. The general regression network (GRN) has been shown to predict acceptable motion simulations for a specific task using few training cases (in the range of tens) [25]. The GRN can simulate the motion of a complete DHM model under various conditions, different loads, and anthropometric variables.

However, the prediction capability of the GRN is limited, because it is susceptible to adverse effects from local optima and poor heuristic settings [26, 27]. A new RBN-based network, called Opt_RBN, has been introduced recently for improved performance with limited available training cases [2]. Opt_RBN entails multi-stage training approaches that produce more objective settings of the network parameters. In addition, it provides improved efficiency with limited training data.

The goals of this work are to: 1) introduce a modified Opt_RBN training process that is capable of being trained for large-scale problems with minimal training data, and 2) apply the new modified Opt_RBN design for real-time prediction of PD tasks for a full-human model. The proposed modified algorithms are tested with two PD tasks: walking and going prone. Although this work presents modification to the Opt_RBN driven by its potential use with PD, the consequent ANN design can be used with a broad range of large-scale problems; PD is simply a well-studied example problem for the proposed developments.

2. BACKGROUND

Predictive dynamics is a physics-based motion simulation algorithm for DHM [1]. The PD method is distinguished from other motion simulation tools, because it produces simulations that reflect the effects of any changes in the DHM conditions, like anthropometric data and loads on the body. This can be particularly useful, as it allows one to study the effects of various conditions on task performance and, for instance, potential injuries.

PD involves solving a nonlinear optimization problem in order to find motion profiles (*i.e.*, control points for B-splines that represent joint-angles over time) for all body DOFs, while adhering to the equation of motion and considering various physical limits and other motion-related constraints. Since its development by [1], PD has been used to simulate different motion tasks and scenarios [28-33]. Successful validation of the PD results has been provided using motion capture systems [34].

The control points that are found by optimization in PD eventually form B-splines for DOFs that simulate the motion in a DHM model. Each DOF has a corresponding B-spline. Having more control points in a B-spline, where their appropriate numbers are determined by the task developer, produces more accurate motion simulation. However, it is computationally more expensive. Although many conditions (*i.e.*, inputs) are common in all tasks, like loading and clothing, others are task specific, like

speed in the walking task, box height in the jumping-on-the-box task, etc. The details of PD settings are provided by [29, 30].

The PD algorithm in this work is applied to the Santos DHM [24]. Santos is a full-body DHM with 109 DOFs. For the purposes of this study 55 DOF are predicted with PD. This work uses Santos with various loading conditions and various limits on range-of-motion. As suggested, creating a relatively large number of simulated motions (from different task conditions) for trade-off analyses can be computationally demanding and thus time consuming. The running time for each case, even with minor changes in initial conditions, can take minutes to hours to complete and produce the simulation. Not only does this inhibit trade-off analyses, but it inhibits potential training of ANNs. Hence, an ANN is needed that requires only minimal training data.

In addition to the challenging concerning run time, PD problems are relatively large with regards to the number of outputs (control points). The number of outputs in a PD task is approximately 500-700, depending on the number of control points in each task. Therefore, running a PD algorithm to create new simulations, either for direct use or for use as one of many training cases, requires the use of special optimization software that is designed to solve large-scale optimization problems. If the Opt_RBN design is used to simulate a PD task, one stage of the network training process that includes solving the optimization problem will not work due to the largescale of the problem (as will be shown in Section 3.1). Thus, new refinements are proposed and tested to resolve this issue.

In the following section, after the fundamental design of the proposed Opt_RBN is presented, new specific methodologies are developed to guarantee successful running of its training process. The consequent modified Opt_RBN design produces accurate and real-time simulation of the large-scale PD problem and can be trained with any optimization algorithm on any PC.

3. METHOD

3.1. Fundamental Design

The fundamental model in this work is based on the new RBN (Opt_RBN) for reduced training sets [2]. Opt_RBNis chosen because it outperforms other typical ANN models, especially for applications with a limited amount of training data. However, as stated earlier, the new Opt_RBN design experiences a memory issue when being trained to predict a relatively large number of outputs [2]. In the case of the PD problem, the original Opt_RBN design cannot be trained when the problem has more than 200 outputs. Hence, for successful training and simulation of the PD problem, this work illustrates modifications to the training process that resolve this issue. The modifications basically allow the training process to be performed by typical algorithms and software, especially for the optimization step of the process, without the need for a special large-scale optimization program or a computer with special processing capabilities. Although the modification to the Opt_RBN is driven by the use of PD, the consequent ANN design can be used for other similar large-scale problems with reduced training sets.

Before it can be used, any ANN needs to be trained on the task being modeled. A detailed description of the fundamental RBN design used is provided in the literature [2]. In summary, the training process in that design consists of four main steps. First, inputs of training cases are all normalized by standardization in order to rescale all inputs to be within a scaleof -1 to 1. Next, preliminary values for the network basis function (*i.e.*, Gaussian) spreads σ^0 are set for all potential basis functions using the root-mean-square-distance (RMSD). Then, the number of network basis functions Q, their centers U, and preliminary values for connection weights W^o are set usingthe orthogonal least square (OLS) approach. Finally, optimal values for W and σ are calculated based on the formulation in Equation 1. This optimization problem incorporates the network parameters (W) as additional design variables. Equation 1 represents a problem with multiple outputs (N).

Minimize: $$f(\sigma,W)=\frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M}\left(t_{nm}-y_{nm}\right)^2 \qquad (1)$$

$$=\frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M}\left(t_{nm}-\left(\sum_{q=1}^{Q}h_{mq}*w_{nq}\right)\right)^2$$

Subject to: $0<\sigma_q\,;q\in R^Q$

In Equation 1, $W=[w_1\,w_2\,...w_N]$ is a matrix of output weight vectors for all N outputs. Each vector $W_n=[w_1\,w_2\,...w_Q]^T$ represents the connections between all Q basis functions with the n^{th} output; t_{nm} is the true m^{th} training value of the n^{th} network output;

y_{nm} is the predicted m^{th} training value of the n^{th} network output; h_{mq} (Equation 2) is the q^{th} basis function output for the m^{th} training case; and w_{nq} is the output connection weight between the q^{th} basis function and n^{th} output.

$$h_{mq} = \exp\left[\frac{-\| x - u_{mq} \|^2}{2\sigma\frac{2}{q}}\right] \quad (2)$$

h_{mq} is inversely proportional to the distance between network's input $x\left(\in R^1\right)$ and u_{mq} which is the q^{th} basis-function center for the m^{th} training case. h_{qm} equals 1 at its maximum, when x equal u_{mq}, and it equals zero at its minimum. The function width σ_q (the Gaussian spread) is the primary tuning parameter with a RBN, along with the number of basis functions (*i.e.*, neurons, Q) in the hidden layer.

With respect to the aforementioned Opt_RBN training process, the optimization step has tens of thousands of design variables (approximately 40,000 on average) when applied to PD problems [2]. Thus, typical CPU memory cannot handle this matrix size all at once, and optimization algorithms cannot solve this large problem. Therefore, the proposed enhancements for the Opt_RBN design in this paper focus on the steps of setting the preliminary basis function spreads σ^0 and calculating the optimal basis function spreads and connection weights ***W***. These modifications are necessary to let the division of the original optimization step be performed in multiple runs for various groups of outputs. Each consequent optimization run has a much smaller problem size (approximately 300-1,000 design variables) than in the original optimization problem (Equation 1). The aggregate solution provided by the proposed multiple optimization problems is exactly the same as the solution obtained with the original single optimization problem (as will be shown in Section 3.2.2). The proposed design modifications are illustrated next.

3.2. New Approaches for Opt_RBN Training Process

3.2.1. Setting Basis Function Spreads

In the original Opt_RBN design, along with the output weights $\left(W = [w_1\, w_2 \ldots w_N]\right)$ corresponding to *N* outputs, the vector of the basis function spreads, σ, is included as a design variable in the optimization step (Equation 1). Unlike the weights, the spreads σ are coupled and cannot be separated for different optimization runs. In other words, all σ elements contribute in producing all network outputs, while each vector of W contributes in producing one of the network outputs independently from the other vectors. That said, in order to be able to formulate multiple smaller optimization problems, σ needs to be removed from the optimization problem as a vector of design variables.

Removing σ from the optimization necessitates introducing more rigorous setting of its preliminary values σ^0, because these values will be used as the final values and will no longer be optimized. This new approach is especially important for an application with either too limited or too many training cases, in order to avoid any excessively large or small spread values. As stated, in the original design, σ^0 elements are set by using RMSD between the normalized inputs of the neighboring training points. However, the RMSD can be approximately zero when there is a relatively large number of training cases, because the training points are close to each other which can produce too small RMSD. When only minimal training cases exist, the calculated σ^0 based on the RMSD can be too large, which produces a network with less accurate results. If the spreads are removed from the optimization step, their values need to be determined using a more intelligent method compared to that used to set their preliminary values in the Opt_RBN, in order to use these values in the final network design. Therefore, a new approach for calculating σ is shown in Equation 3.

$$\sigma_j = \left(\frac{M}{I}\right)\| u_{jk} \| = \frac{M}{I}\sum_{i=1}^{I}\left|u_{ji} - u_{j,ki}\right|, \quad j \in \mathbb{R}^M \quad (3)$$

Setting σ for large-scale problems depends on the Manhattan distance $(\| u_{jk} \|_1)$ [35, 36] between the input vectors of two adjacent training cases and the ratio of the number of training cases to the number of inputs $\left(\frac{M}{I}\right)$ This step assumes all available training cases are used as potential basis functions [2]. Thus, a spread value is set for each available training case. Then, the spreads that correspond to the selected training cases, which are essentially the basis functions selected by the OLS approach, are used in the final network design.

In Equation 3, σ_j is the spread of the j^{th} training case when its input is selected as the basis function. I is the size of input vector (*i.e.*, the number of elements in the input vector). M is the number of training cases.

$\|u_{jk}\|_1$ represents the Manhattan distance between the inputs of the j^{th} training case and its closest K^{th} training cases. u_{ji} is the i^{th} input element of the j^{th} training case. $u_{j,ki}$ is the i^{th} input element of the K^{th} training case that is closest to the j^{th} training case.

In this context, the Manhattan distance ($\|u_{jk}\|_1$) is the sum of absolute distances between two points (vectors) in all dimensions. Note that in a higher dimensional space, this distance provides more accurate representation of the actual distance between two points compared to that provided by the Euclidean distance. With respect to the ratio $\left(\frac{M}{I}\right)$, it accounts for the training size and input dimension of the application in order to provide appropriate σ values for any problem. Preliminary work was performed by investigating multiple approaches for various ratios based on the network and problem parameters that lead to the ratio $\frac{M}{I}$ as an acceptable heuristic setting. In general, if the input size (*i.e.*, the number of input parameters) I increases, more training cases M are required to provide more combinations of these parameters. The increase in I is implicitly considered by increasing M. Even when I increases with fixed M, the resulting $\|u_{jk}\|_1$ is larger, because typically the training points are further from each other in higher dimensional space. The larger $\|u_{jk}\|_1$ compensates for the fixed M producing larger σ_j. Hence, the ratio $\frac{M}{I}$. provides an appropriate factor for any problem to produce proper $\boldsymbol{\sigma}$ values in order to handle a problem of any size. In addition, this ratio reduces the propensity for prohibitively small $\boldsymbol{\sigma}$ values. Even when the problem has a relatively large number of training cases (in the hundreds or thousands), the $\boldsymbol{\sigma}$ values will not be too small, because the possible too small $\|u_{jk}\|_1$ value is multiplied by large ratio $\frac{M}{I}$.

The heuristic-based σ values in the new method are used as the final values. The new method is appropriate for setting a value σ that depends on the $\|u_{jk}\|_1$ value and various combinations of training cases and inputs.

3.2.2. Grouped Optimization of Network Output Weights

After the spreads σ are removed as design variables from the original optimization problem, the weights W remain as the only design variables. The weights $W = [w_1 w_2 ... w_N]$ for different outputs can then be decoupled, because each vector of output weights $W_n = [w_1, w_2, ... w_Q]^T$ is independent from the others used to produce the network outputs (as shown in Equation 1). Thus, this section details a new *grouped optimization* step for each group of outputs that are related to each other (Equation 4). Multiple grouped optimization runs are then performed, and each includes the weights corresponding to the outputs that are related to 1 DOF (*i.e.*, each group of outputs denotes the control points of 1 DOF in the PD problem). There are 55 separate optimization runs corresponding to the 55 predicted DOFs of the DHM. In contrast to the original optimization problem, which involves finding all vectors of W in a single run, the new optimization involves multiple runs, each for a smaller group of vectors W_g The new optimization is also simpler compared to that in Equation 1, because it is unconstrained. Note that since each vector in W is independent from the others, Equation 4 can be generalized and rewritten to suite any problem with any number of outputs.

Find: $W_g = [w_1 w_2 ... w_D]; g = 1,2,..., G$

Minimize: $f(W_g) = \frac{1}{D}\sum_{d=1}^{D}\sum_{m=1}^{M}\left(t_{g,d,m} - y_{g,d,m}\right)^2$

$$= \frac{1}{D}\sum_{d=1}^{D}\sum_{m=1}^{M}\left(t_{g,d,m} - \sum_{q=1}^{Q}\left[h_{m,q} * w_{g,d,q}\right]\right)^2 \quad (4)$$

In Equation 4, G is the number of optimization problems to be solved (55). $W_g = [w_{g,1}, w_{g,2}, ... w_{g,D}]$ is the connection-weight matrix that corresponds to the g^{th} group of outputs $\left(W_g \in R^D\right)$. $W_{g,d} = [w_{g,d,1}, w_{g,d,2}, ... w_{g,d,Q}]^T$ is the connection-weight vector that corresponds to the d^{th} output in the g^{th} group. *D* is the number of outputs in each g^{th} group. $t_{g,d,m}$ is the true value for the d^{th} output of the m^{th} training case in the g^{th} group. $y_{g,d,m}$ is the predicted (network) value for the d^{th} output of the m^{th} training case in the g^{th} group. $h_{m,q}$ is the q^{th} basis function output when considering the input of the m^{th} training case $w_{g,d,q}$ is the connection-weight value between the q^{th} basis function and the d^{th} output in the g^{th} group of optimization.

The memory and run-time issues are resolved by using this new grouped optimization approach. As an example to demonstrate that, a comparison of the running time is performed between the single optimization and the grouped optimization for the PD problem. In this case, the network is trained to simulate

part of the PD outputs, since 200 outputs is the maximum number of outputs the optimization can solve in a single run as opposed to approximately 500 to 700 outputs in a full-PD problem. For a PD problem with 200 outputs, the running time in the new grouped optimization is approximately 20 minutes, while it is approximately 24 minutes for the single optimization (the optimization is solved on a Windows 7 computer with an Intel® Core™ i7 processor and 16 GB of RAM). The solutions from both problems are exactly the same, which reassures the independencies among W vectors.

4. RESULTS

The new modified Opt_RBN is tested using two PD task-simulations that are created for the Santos software. The tasks are walking forward and going prone. These tasks are selected, because they differ in terms of complexity and behavior. The modified Opt_RBN performance is evaluated objectively by calculating test error and comparing the error with the results of a typical RBN design [37, 38]. In addition, subjective evaluation of the visual results is provided.

4.1. Example 1: Walking Task

The walking task is a common and basic task often discussed in the field of DHM. In this work, the task includes 42 inputs, which represent the loading conditions (16 inputs), joint ranges of motion (24 inputs), weapon point-location on the hands (1 input), and walking speed (1 input). The loading conditions include the total weight of the added equipment on the back, and the body-segment centers-of-mass in three dimensions. The ranges of motion (ROM) include normal and reduced ranges for the bending, extension-flexion, and rotation for four of the DHM's spinal joints. Although other ROMs can be included as extra inputs in the task, this work includes these spinal ROMs specifically to evaluate the effects of various loadings, which are mainly added on the back, and on the spinal joints. The task has 9 control points for each DOF, which translates into a total of 495 outputs. 399 training cases are collected, representing various combinations of inputs. The network training time is approximately 41 minutes. The final network includes 74 basis functions (*i.e.*, neurons).

To evaluate the network performance, five test cases are used. The test cases are cases that have never been used to train the network. With regard to objective evaluation, the root mean square error (RMSE) for the modified Opt_RBN predicted outputs are compared with those produced from RBN, which takes approximately 6 minutes to be trained for this task. The average RMSE for the five test cases is 0.031 for the Opt_RBN and 0.04 for the RBN. Although the results are small for both models, the Opt_RBN has approximately 25% less error. Since the outputs are calculated in radians, which are the reason for obtaining apparently small RMSE in both models, direct conversion to degrees yields an error of approximately 1.78 for the Opt_RBN and 2.3 for the RBN. When predicting joint angles, reducing the error by 25%, on average, is a significant improvement, because even errors as small as 2.3 degrees in each joint angle profile can result in odd visual motion simulations.

Another objective measurement of the results' accuracy is the number of outputs each of the network models (Opt_RBN and RBN) can predict with less error than the other model (*i.e.*, count the number of outputs with smaller RMSE for each network). Since the PD problem involves hundreds of outputs, this measurement is helpful for evaluating the performance of the Opt_RBN design. Such a measurement can be helpful in studying the general trend of the network when predicting each output.

The average number of the more accurately predicted outputs among the five test cases is calculated. The calculated results show that the Opt_RBN is more accurate than RBN in this task with an average of 269 outputs compared to 217 for the RBN. Both networks have the same exact error in 9 outputs. Among the 486 compared outputs, the Opt_RBN provides less error in 52 more outputs than the RBN. The performed objective evaluations are based on the reported RMSEs.

With respect to the subjective evaluation, the simulation results from the Opt_RBN design are evaluated visually. The RBN visual results are not included, because the visual results from both models do not show obvious difference over the whole motion. Figure **1** shows the visual results for the Opt_RBN, where the motions resulting from the five test cases are simulated for the Santos model. In the figure, Santos moves from right to left. All of the simulated motions produced from the Opt_RBN look acceptable, suggesting the network is generally able to predict proper simulations for the corresponding input conditions. Specifically, the effect of heavier loads is represented in all simulations.

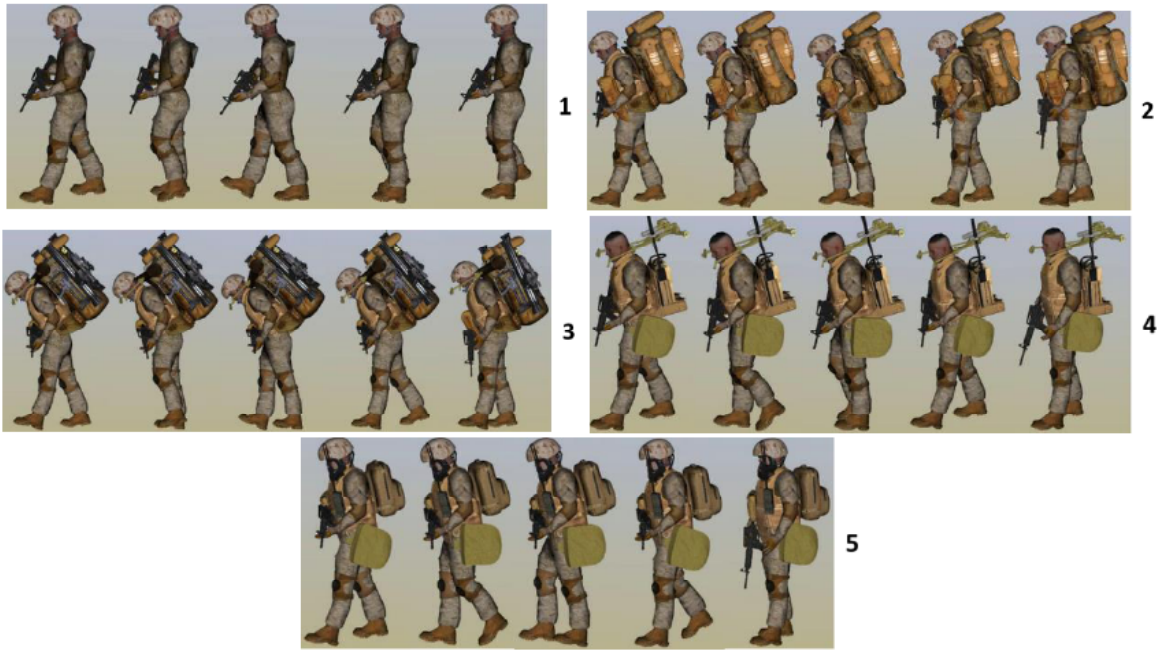

Figure 1: Selected key frames for walking task simulation results of test cases **1-5** using the modified Opt_RBN.

In summary, with the walking task, the Opt_RBN performance is successful in providing high-fidelity outputs. The small RMSE values for the presented test cases show the accuracy of the results obtained from the Opt_RBN design. The visual evaluation also illustrates acceptable results.

4.2. Example 2: Going-Prone Task

Going prone is another task that is commonly performed by today's War fighter. The task has 41 inputs, which represent the loading conditions (16 inputs), joint ROMs (24 inputs), and weapon point-location on the hands (1 input). The task has the same inputs used for the walking task, with the exception of walking speed. There are 550 outputs, since there are 10 control points for each DOF. The task involves 306 training cases. The training process takes approximately 18 minutes, and the final network includes 38 basis functions. Five test cases, which represent five different loading and ROM conditions, are evaluated. The same subjective and objective evaluations are performed on the modified Opt_RBN results.

For the results of predicting the five test cases, the average RMSE for Opt_RBN predicted outputs is 0.018, while it is 0.026 for RBN, which takes approximately 4 minutes to be trained for this task. As with the walking task, when the prediction errors are converted to degrees, the RMSE is 1.03 for the Opt_RBN and 1.5 for RBN. This comparison shows an approximately 30% improvement for the results produced by Opt_RBN as compared to those from the RBN. As mentioned, the improvement of seemingly small errors in the PD task is critical, because of the necessity for the highest possible accuracy in the nature of the simulated PD problems. Reducing the average error for each DOF from 1.5 to 1 degree can produce a significant difference when these errors compound for the full 55-DOF DHM. In general, although the simulated motions in many cases produced from the Opt_RBN and RBN might not be visually notable or differ significantly, these differences could occur in many other cases, especially in cases where prediction errors are present in most of the DOFs. In addition, in complicated problems like the PD tasks, where there are hundreds to thousands of constraints to be met, any improvements in the simulated motions, even with seemingly fewer errors, means fewer violated constraints.

In terms of model comparison based on the number of outputs with smaller errors, the Opt_RBN design shows superiority over the RBN with an average count of 181 for the Opt_RBN versus 172 for the RBN. Even though the numbers are close, this result provides further confirmation that the RBN does not outperform Opt_RBN by any means. The better RMSE value presented for Opt_RBN than for RBN supports that conclusion.

Another significant result regarding the going-prone task is the fact that both networks produce the same error values in 197 outputs. This relatively large number of outputs with the same error values demonstrates that there is tight motion variability in the going-prone task, and the task shows fewer effects on the resulting motion over various changes in loading conditions compared to the walking task. This in turn means that the task has some joint DOFs, which are represented as a group of control points, with minor changes at various task conditions. Consequently, both network models are able to predict these 197 slightly changed outputs with the same high accuracy level.

With respect to the subjective evaluation, the simulation results from the Opt_RBN design are evaluated visually for the five test cases of the going-prone task. Figure **2** provides visual representation for the simulation results produced from the Opt_RBN. Again, all simulated motions are acceptable. With fewer effects of various loads on the resulting motion than in the walking task, this task shows a unique strategy for handling different load configurations in each case. In general, the Opt_RBN is successful in providing high-fidelity results for the going-prone task, which involves more outputs to predict and fewer available training cases than the walking task.

From both objective and subjective perspectives, the performance of the new Opt_RBN in predicting the going-prone task is acceptable. The new design outperforms RBN when they are compared in terms of the RMSE and in terms of the number of outputs with lower error values. As with the first task, the visual evaluation of the produced motion (*i.e.*, outputs) from Opt-RBN is acceptable in all cases.

4.3. Sensitivity Analysis

A critical aspect of any ANN is its sensitivity to the amount of training. A primary novelty for the proposed work is its need for just minimal training; it is effective even when extensive training data is not available. To check the sensitivity of Opt_RBN accuracy to the numbers of training cases, it is trained and evaluated with various numbers of cases for a walking task, and its results are compared with those obtained from the RBN.

The numbers of training cases used in this study are 44, 132, 198, 399, 918, 1224, and 1529. The average error results of predicting five test cases for both networks are presented in Figure **3**. The modified Opt_RBN clearly outperforms the RBN. For example, when both networks are trained with 132 cases, the error produced from Opt_RBN is 20% less than that in RBN, and slightly better than that produced from RBN when it is trained with 198 cases. Similar results are produced from both networks when Opt_RBN is trained with 198 cases and RBN is trained with 918 cases. These results show that the Opt_RBN can use as few as 25% of the training cases required by the RBN to produce the same prediction errors. With problems like PD, where training data may be limited, the insensitivity to training data is critical.

In Figure **3**, although the minimum RMSE that is reached by RBN and Opt_RBN when they are trained with 1529 cases is not significantly different (the RBN error is 0.031 and Opt_RBN is 0.027), the results indicate the superiority of the Opt_RBN over the RBN even when trained with a larger number of training cases.

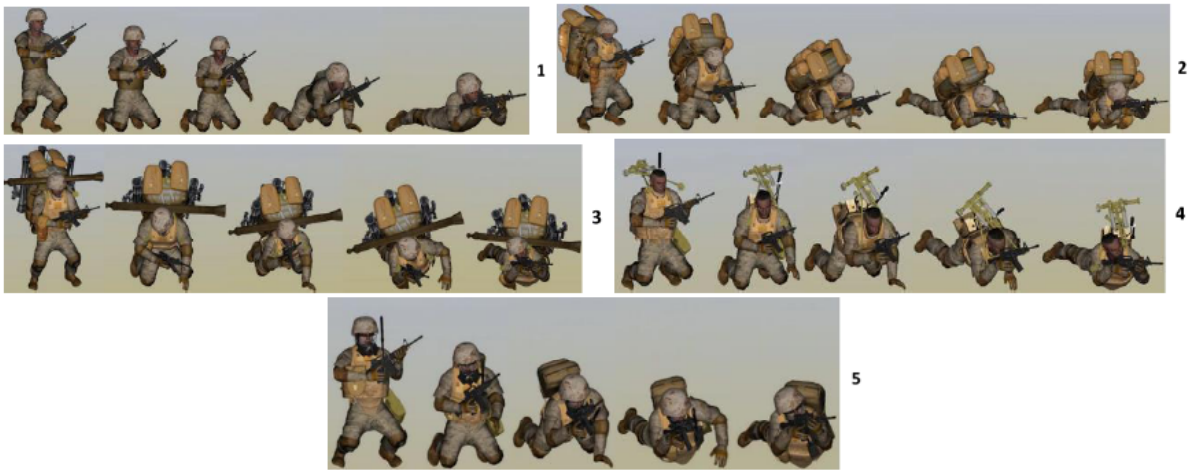

Figure 2: Selected key frames for going-prone task simulation results of test cases 1-5 using the modified Opt_RBN.

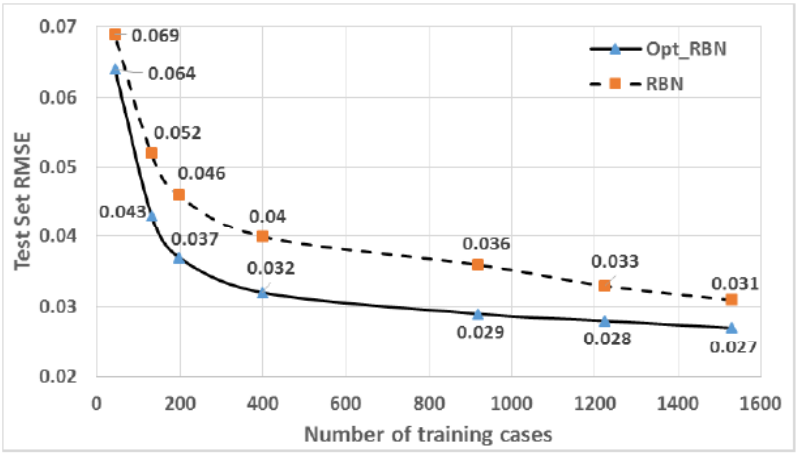

Figure 3: Test set RMSE evaluation for the modified Opt_RBN and typical RBN at various numbers of training cases when simulating the walking task.

When Opt_RBN is used to predict a PD task, the presented error analysis can provide an approximation of the number of training cases needed for the network training process to produce a model with acceptable accuracy. The results in Figure **3** show that, after the Opt_RBN is trained with 918 cases, no significant improvement is obtained. In addition, given the large number of cases added to train the network, and keeping in mind the necessity of training with a minimal set of cases, the error reduction from 0.029 to 0.027 is negligible. The reduction in the number of training cases in the PD application is critical for saving effort, since collecting each PD case is computationally costly.

5. DISCUSSIONS AND CONCLUSION

This work presents a new ANN that is designed for large-scale problems with minimal data sets. As so called, big data problems continue to surface and applications of artificial intelligence and machine learning accelerate, this new network promises far reaching benefits. Although this work leverages the recently introduced RBN design, called Opt_RBN, the enhancements are novel. The original design, which is intended for applications with a reduced number of training cases, is modified to work for large-scale problems in terms of the number of outputs. Optimization-based prediction of dynamic human motion is used as an initial, yet challenging example problem. To be sure, future work includes additional large problems in various fields

The initial Opt_RBN is proven to improve the prediction results for applications with a reduced number of training cases. However, applying this network to the large-scale PD application experiences some difficulty. Specifically, the Opt_RBN experiences a CPU memory issue when solving the internal optimization problem for the training process. With the successful implementation of the modified algorithms within the Opt_RBN design in this work, the network' scapability for real-time simulation of a PD task is achieved and all memory issues are resolved. The proposed network is tested successfully with two tasks (walking and going prone). In general, the results of the new network are validated objectively and subjectively. The objective comparisons for the Opt_RBN results with those from the RBN show superior performance by the Opt_RBN design. The reported RMSEs for both networks indicate more than 25% improvement of the new network for both presented tasks.

In summary, the main contributions of this work are as follows:

1. Development of a modified Opt_RBN training process for use with large-scale problems and minimal training data.

2. Application of the new modified Opt_RBN design for real-time prediction of dynamic human motion.

Although this work presents modification to the Opt_RBN driven by its potential use with PD, the consequent ANN design can be used with a broad range of large-scale problems.

RBN took approximately 2 to 3 minutes to train for each task, whereas the new network took approximately 18-40 minutes. Both networks run in a fraction of a second for the test cases. Given the

improvements in the results and the problem sizes, the training times for the new network are acceptable. Furthermore, the training time is not as important as the run time for test cases for most practical applications.

Sensitivity analysis is performed for the modified Opt_RBN and compared with the RBN to check the performance at various numbers of training cases. Besides providing additional evidence of the superiority of Opt_RBN's performance over RBN's, the analysis introduces a general tool that can be used to guide collecting the proper number of training cases. It can be useful with other applications to evaluate what level of training is necessary for a desired level of accuracy.

A significant consideration in the field of machine learning is the training of the various algorithms. Despite increasing media reports of the power of artificial intelligence, any cognitive capability- either it is human or mathematical- requires training. This paper indirectly raises the issue of advantages gained from faster and more efficient training. Considering the human analogy, large sets of available data can essentially accelerate the training that occurs over one's life. ANNs need not exist for a generation in order to gain "wisdom". The data of a lifetime is already available in many instances. However, the proposed algorithm is a step towards using this data more efficiently.

In addition to providing a new and useful ANN, this work has flushed out directions for future work. It will be beneficial to investigate finding a proper ratio of outputs to inputs and/or number of training cases for improved ANN performance in PD applications. Furthermore, based on the presented results for the evaluated PD tasks, which show that some task outputs have either minor changes or no changes at all over various task conditions, the ANN can help in developing new tools to improve the PD task development process. For example, when a PD task is developed, the task could have a reduced number of design variables (*i.e.*, eliminate the unchanged outputs) or could involve fixing certain DOF. That in turn would save development time and computational run time. Future work also entails additional applications for the proposed network.

ACKNOWLEDGEMENT

This work is funded by the US Office of Naval Research, and was performed at the University of Iowa's Center for Computer Aided Design, where the authors worked at the time. The authors would like to thank the team at the University of Iowa's Virtual Soldier Research (VSR) program.

REFERENCES

[1] Xiang Y, Chung H-J, Kim JH, Bhatt R, Rahmatalla S, Yang J, *et al*. Predictive dynamics: an optimization-based novel approach for human motion simulation. Structural and Multidisciplinary Optimization. 2010; 41(3): 465. https://doi.org/10.1007/s00158-009-0423-z

[2] Bataineh M. New neural network for real-time human dynamic motion prediction.PhD thesis, University of Iowa 2015.

[3] Collobert R, Bengio S, Bengio Y. A parallel mixture of SVMs for very large scale problems. Neural Comput 2002; 14(5): 1105-14. https://doi.org/10.1162/089976602753633402

[4] Dean J, Corrado G, Monga R, Chen K, Devin M, Mao M, *et al*., editors. Large scale distributed deep networks. Advances in Neural Information Processing Systems; 2012.

[5] Karpathy A, Toderici G, Shetty S, Leung T, Sukthankar R, Fei-Fei L, editors. Large-scale video classification with convolutional neural networks. Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on; 2014: IEEE.

[6] Bottou L. Large-scale machine learning with stochastic gradient descent. Proceedings of COMPSTAT'2010: Springer; 2010. p. 177-86. https://doi.org/10.1007/978-3-7908-2604-3_16

[7] Rao R, Savsani V, Vakharia D. Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems. Inf Sci 2012; 183(1): 1-15. https://doi.org/10.1016/j.ins.2011.08.006

[8] Eliasmith C, Stewart TC, Choo X, Bekolay T, DeWolf T, Tang Y, *et al*. A large-scale model of the functioning brain science 2012; 338(6111): 1202-5.

[9] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv: 14091556. 2014.

[10] Arndt O, Barth T, Freisleben B, Grauer M. Approximating a finite element model by neural network prediction for facility optimization in groundwater engineering. European journal of operational research 2005; 166(3): 769-81. https://doi.org/10.1016/j.ejor.2003.09.039

[11] Zadpoor AA, Campoli G, Weinans H. Neural network prediction of load from the morphology of trabecular bone. Applied Mathematical Modelling 2013; 37(7): 5260-76. https://doi.org/10.1016/j.apm.2012.10.049

[12] Leighton FT. Introduction to parallel algorithms and architectures: Arrays· trees· hypercubes: Elsevier; 2014.

[13] Boyd S, Parikh N, Chu E, Peleato B, Eckstein J. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine Learning 2011; 3(1): 1-122. https://doi.org/10.1561/2200000016

[14] Garland M, Le Grand S, Nickolls J, Anderson J, Hardwick J, Morton S, *et al*. Parallel computing experiences with CUDA. IEEE micro 2008(4): 13-27. https://doi.org/10.1109/MM.2008.57

[15] Ciresan DC, Meier U, Gambardella LM, Schmidhuber J. Deep, big, simple neural nets for handwritten digit recognition. Neural Comput 2010; 22(12): 3207-20. https://doi.org/10.1162/NECO_a_00052

[16] Krizhevsky A, Sutskever I, Hinton GE, editors. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems; 2012.

[17] Ciresan D, Meier U, Schmidhuber J, editors. Multi-column deep neural networks for image classification. Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on; 2012: IEEE.

[18] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, *et al*. Going deeper with convolutions. arXiv preprint arXiv: 14094842. 2014.

[19] Bousquet O, Bottou L, editors. The tradeoffs of large scale learning. Advances in neural information processing systems; 2008.

[20] Chalodhorn R, MacDorman KF, Asada M. Humanoid robot motion recognition and reproduction. Advanced Robotics 2009; 23(3): 349-66. https://doi.org/10.1163/156855308X397569

[21] Calinon S, Guenter F, Billard A. On learning, representing, and generalizing a task in a humanoid robot. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 2007; 37(2): 286-98. https://doi.org/10.1109/TSMCB.2006.886952

[22] Bu N, Okamoto M, Tsuji T. A hybrid motion classification approach for EMG-based human–robot interfaces using bayesian and neural networks. Robotics, IEEE Transactions on 2009; 25(3): 502-11. https://doi.org/10.1109/TRO.2009.2019782

[23] Reinhart RF, Steil JJ, editors. Reaching movement generation with a recurrent neural network based on learning inverse kinematics for the humanoid robot iCub. Humanoid Robots, 2009 Humanoids 2009 9th IEEE-RAS International Conference on 2009: IEEE.

[24] Abdel-Malek K, Yang J, Kim JH, Marler T, Beck S, Swan C, *et al*. Development of the virtual-human SantosTM. Digital Human Modeling: Springer 2007; 490-9. https://doi.org/10.1007/978-3-540-73321-8_57

[25] Bataineh M, Marler T, Abdel-Malek K, Arora J. Neural network for dynamic human motion prediction. Expert Systems with Applications 2015.

[26] Masters T, Land W, editors. A new training algorithm for the general regression neural network. Systems, Man, and Cybernetics, 1997 Computational Cybernetics and Simulation, 1997 IEEE International Conference on 1997: IEEE.

[27] Tomandl D, Schober A. A modified general regression neural network (MGRNN) with new, efficient training algorithms as a robust 'black box'-tool for data analysis.Neural Networks 2001; 14(8): 1023-34. https://doi.org/10.1016/S0893-6080(01)00051-X

[28] Xiang Y, Arora JS, Rahmatalla S, Marler T, Bhatt R, Abdel-Malek K. Human lifting simulation using a multi-objective optimization approach. Multibody System Dynamics 2010; 23(4): 431-51. https://doi.org/10.1007/s11044-009-9186-y

[29] Xiang Y, Arora JS, Abdel-Malek K. Physics-based modeling and simulation of human walking: a review of optimization-based and other approaches. Structural and Multidisciplinary Optimization 2010; 42(1): 1-23. https://doi.org/10.1007/s00158-010-0496-8

[30] Xiang Y, Arora JS, Abdel-Malek K. Hybrid predictive dynamics: a new approach to simulate human motion. Multibody System Dynamics 2012; 28(3): 199-224. https://doi.org/10.1007/s11044-012-9306-y

[31] Kwon H-J, Xiang Y, Bhatt R, Rahmatalla S, Arora JS, Abdel-Malek K. Backward walking simulation of humans using optimization. Structural and Multidisciplinary Optimization. 2014: 1-11. https://doi.org/10.1007/s00158-013-1039-x

[32] Kim JH, Xiang Y, Bhatt R, Yang J, Chung H-J, Patrick A, *et al*., editors. Efficient ZMP formulation and effective whole-body motion generation for a human-like mechanism. ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference; 2008: American Society of Mechanical Engineers.

[33] Kim JH, Xiang Y, Yang J, Arora JS, Abdel-Malek K. Dynamic motion planning of overarm throw for a biped human multibody system. Multibody System Dynamics 2010; 24(1): 1-24. https://doi.org/10.1007/s11044-010-9193-z

[34] Rahmatalla S, Xiang Y, Smith R, Meusch J, Bhatt R. A validation framework for predictive human models.International journal of human factors modelling and simulation 2011; 2(1): 67-84. https://doi.org/10.1504/IJHFMS.2011.041638

[35] Köthe G, Garling DJH. Topological vector spaces: Springer 1969.

[36] Schaefer HH, Wolff MP. Locally Convex Topological Vector Spaces: Springer; 1999. https://doi.org/10.1007/978-1-4612-1468-7

[37] Beale MH, Hagan MT, Demuth HB. Neural network toolbox for use with Matlab user's guide version 4.The Mathworks 2001.

[38] Chen S, Cowan CFN, Grant PM. Orthogonal least squares learning algorithm for radial basis function networks. Neural Networks, IEEE Transactions on 1991; 2(2): 302. https://doi.org/10.1109/72.80341

Received on 29-10-2016 Accepted on 28-11-2016 Published on 24-12-2016

http://dx.doi.org/10.15379/2410-2938.2016.03.02.02