A Hybrid Training Algorithm for Recurrent Neural Network Using Particle Swarm Optimization-based Preprocessing and Temporal Error Aggregation

Qiao Kang, Wei-keng Liao, Ankit Agrawal, and Alok Choudhary
EECS Department, Northwestern University
Email: {qiao.kang, wkliao, ankitag, choudhar}@eecs.northwestern.edu

***Abstract*—Recurrent neural network has been widely used as auto-regressive model for time series. The most commonly used training method for recurrent neural network is back propagation. However, recurrent neural networks trained with back propagation can get trapped at local minima and saddle points. In these cases, auto-regressive models cannot effectively model time series patterns. In order to address these problems, we propose a hybrid recurrent neural network training algorithm that consists of two phases: exploration and exploitation. Exploration phase uses synchronous particle swarm optimization to search for parameter settings with high activation score and low error. The results of exploration phase are trained with proposed enhanced back propagation, an improved algorithm over traditional back propagation that aggregates temporal errors across timestamps, in exploitation phase. We evaluate our proposed methods using four real-world datasets. Our proposed algorithm, applied to both regularized and adaptive momentum back propagation, increases convergence speed by 10% to 20% and reduces testing mean square error(MSE) at convergence by 5% to 30%. Using particle swarm optimization and activation list in exploration phase, the hybrid training algorithm reduces testing MSEs by more than 30% at convergence compared with traditional back propagation.**

I. Introduction

Elman recurrent neural network[1] has been widely used as an auto-regressive (AR) model in many applications. For example, Hajdarevic uses recurrent neural network to detect anomalies in thermal plant[2]. Xiao uses recurrent neural network for impedance identification[3]. Recurrent neural network transits its internal state every time it processes input. This property makes it suitable for modeling complex patterns in time series.

Training recurrent neural network has two important performance considerations, namely final error and convergence speed. Increasing convergence speed allows building of models on larger datasets with large numbers of parameters. Reducing errors improves model prediction accuracy, thereby making the recurrent neural network more actionable. Back propagation is a dynamic programming design of first order gradient descent for training neural network [4]. Most recurrent neural networks are non-convex functions, so the final solution of gradient descent can be trapped at local minima or saddle points[5]. The current solution to obviate local minima is to train multiple networks with randomized initial weights and select the one with the lowest errors, but the execution time can become prohibitively expensive. Thus, techniques that can improve both the convergence speed and accuracy are needed.

To achieve the goal of faster convergence in back propagation, we propose an enhanced back propagation algorithm that aggregates averaged errors from future timestamps. A dual equation system is used to delay error averaging in order to solve numerical overflow problem without losing error size. To reduce the final error, we present a new training method that is based on the concept of particle swarm optimization (PSO)[6] for finding the best-fit initial parameters. Such initial parameters enable the proposed back propagation algorithm to produce results with much lower errors. The proposed hybrid algorithm consists of two phases: **exploration** and **exploitation**. Firstly, PSO is used to explore the search space of parameters based on the activation score. During exploration, an activation list is maintained to store the candidate parameters that produces lowest errors and highest activation scores. Activation score is computed using the principles of network weights initialization method as proposed in [7]. A network with high activation score implies low convergence error after training with back propagation. At exploitation stage, all parameter settings for the network in the activation list are trained with enhanced back propagation algorithm. The network that yields the lowest final error is returned as the final output.

We evaluate our proposed methods using four real-world time series datasets: the QT ECG dataset[8] containing ECG curves, MGH/MF dataset[9] recording three-dimensional heart pressures, and two NASA space shuttle datasets[10] collecting activities of space shuttle's Marotta Valve. Enhanced back propagation, applied to both regularized and adaptive momentum back propagation, increases convergence speed by 10% to 20% and reduces testing MSE at convergence by 5% to 30%. Using PSO and activation list in exploration phase, our hybrid training algorithm reduces testing MSEs by more than 30% at convergence compared with traditional back propagation.

2375-9259/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDMW.2017.112

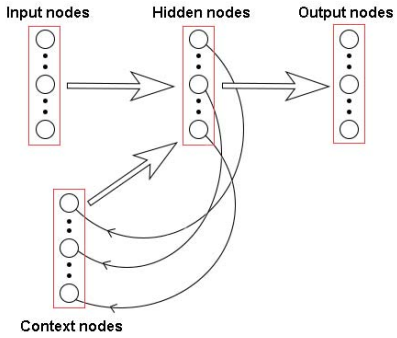


Figure 1: An illustration of Elman neural recurrent network.

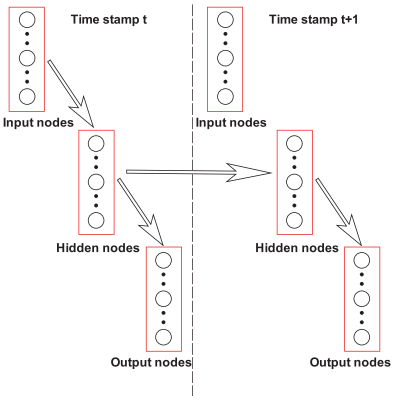


Figure 2: Unfolded Elman neural network.

II. Background and Related Works

Elman neural network, referred as recurrent neural network, uses context layer to adjust its state at different timestamps. Figure 1 illustrates Elman neural network. The white arrows denote full connections between layers. A recurrent neural network is equivalent to a Turing machine[11], so it is capable of encoding complex patterns. There are many variants of recurrent neural networks for addressing specific problems, such as echo state network[12], Jordan network[13], and LSTMs[14].

$$h_t = f(Aa_t + Cc_{t-1}) \tag{1}$$
$$c_t = h_t \tag{2}$$
$$o_t = g(Bh_t) \tag{3}$$

We formulate the notations used in [15] into compact matrix-vector representations. Vectors a_t, h_t, c_t, and o_t are outputs of input layer, hidden layer, context layer and output layer respectively at time stamp t. We denote their size by $|a|$, $|h|$, $|c|$, and $|o|$, where $|c| = |h|$. Let A be a $|h| \times |a|$ matrix such that $A[j,i]$ represents weight from input node i to hidden node j. Let B be a $|o| \times |h|$ matrix such that $B[j,i]$ represents weight from hidden node i to output node j. Let C be a $|h| \times |h|$ matrix such that $C[j,i]$ represents the weight from context node i to hidden node j. Let $f : \mathbb{R}^{|h|} \to \mathbb{R}^{|h|}$ be the activation function at hidden layer and $g : \mathbb{R}^{|o|} \to \mathbb{R}^{|o|}$ be the activation function at output layer. c_t is defined to be a zero vector when $t < 0$. For $t \geq 0$, h_t is defined by Equation 1. c_t is defined by Equation 2. o_t is defined by Equation 3.

A. Back Propagation

Elman neural network can be trained with back propagation algorithm iteratively in four phases. Firstly, the network is unfolded across all time stamps. Figure 2 illustrates an Elman neural network unfolded across two timestamps. The second phase is computing errors. Let Δ_t be a vector of size $|o|$ that represents the output layer errors at time t. Let δ_t be a vector of size $|h|$ that represents the hidden layer errors at time t. Let e_t be the error vector defined by the difference between o_t and actual output of the network at time t. Let f' and g' be the first order derivatives of activation functions f and g. Δ_t and δ_t are computed based on Equations 4 and 5. In the third phase, the gradients for all parameters represented by matrices A, B, and C are computed using Equations 6, 7, and 8. In the final phase, gradients are averaged across all timestamps for gradient descent parameter updates.

$$\Delta_t = g'(Bh_t) \odot e_t \tag{4}$$
$$\delta_t = (B^T \Delta_t) \odot (f'(Aa_t + Cc_{t-1})) \tag{5}$$
$$\frac{de_t}{dB[j,i]} = h_t[i]\Delta_t[j] \tag{6}$$
$$\frac{de_t}{dA[j,i]} = a_t[i]\delta_t[j] \tag{7}$$
$$\frac{de_t}{dC[j,i]} = h_{t-1}[i]\delta_t[j] \tag{8}$$

The calculation of first order gradient descent of neural network, described above, brings up two challenges. The first challenge is to avoid the local optimal solutions caused by non-convexity. The gradient descent process fails to reduce error if a local optimum is reached. A possible solution is to randomly initialize multiple weights parameters and to apply gradient descent to all of them. However, this method has scaling problem because the number of random initialization is proportional to parameter size. Adding a momentum factor during parameter updates is another solution. For example, the ADAM algorithm[16] is designed for varying learning rate at current iteration based on momentums. Recent research has also shown that saddle points can slow down gradient descent convergence[5]. l_2 regularization, which adds penalty for large weights, can relieve the risk of saddle points.

B. Particle Swarm Optimization

Particle swarm optimization (PSO)[6] is generic optimization algorithm for arbitrary objective functions. A particle represents one parameter assignment. In case of recurrent neural network, the parameter assignment includes matrices A, B, and C. PSO allows particles to adjust their locations based on both global and local best locations. The rule of thumb for setting the number of

Algorithm 1: Hybrid Algorithm

Data: Time series *data*
Result: Trained Elman network *weights*
1 $E \leftarrow$ Synchronous PSO
2 $\forall e \in E$ Apply Enhanced Back Propagation to e
3 return $e \in E$ with smallest $MSE(data)$

particles is $p = 3N + 1$, where N is the number of parameters[17]. All particles share the same global best location g_{best}, the location with the smallest MSE. Every particle also has a local best vector l_{best}, the location with smallest MSE the particle traveled. The velocity vector, determined by the displacements from current location to the locations of both local best particle and global best particle stochastically, is used to update every particle.

The idea of combining particle swarm optimization and back propagation has been discussed in [18] and [3]. However, they ignore exploration for a wide range of intermediate results. In [18], there is no guidance for selecting networks from PSO phase other than MSE, so the output of PSO phase may already be local minimum solution. In [3], the authors use gradient as a metric to select outputs from PSO phase. However, large gradient does not necessarily imply lower convergence error after gradient descent. Our proposed hybrid algorithm is different from [3]. Instead of selecting networks from final states of particles, we apply the activation score and the activation list that can thoroughly explore parameter space in the PSO phase. Moreover, we fundamentally improve back propagation algorithm to achieve faster convergence rates with lower errors.

III. Design

The proposed hybrid algorithm consists of two main phases: exploration and exploitation. Exploration phase uses synchronous PSO to update a global activation list and exploitation uses an enhanced back propagation to train output from the exploration phase. Algorithm 1 describes the proposed hybrid algorithm. Line 1 is the exploration phase and line 2 is the exploitation phase. The parameter assignment that yields the smallest MSE is returned at line 3.

A. Exploration Phase

Definition III.1. *The activation score of a recurrent neural network is defined by Equation 9:*

$$1_{|h|}^T I_1(A) 1_{|a|} + 1_{|o|}^T I_2(B) 1_{|h|} + 1_{|h|}^T I_3(C) 1_{|h|} \tag{9}$$

1_c *is a vector if size* c *filled with 1 for* $c \in \mathbb{Z}^+$. $I_1 : \mathbb{R}^{|h|\times|a|} \rightarrow \{0,1\}^{|h|\times|a|}$, $I_2 : \mathbb{R}^{|o|\times|h|} \rightarrow \{0,1\}^{|o|\times|h|}$, *and* $I_3 : \mathbb{R}^{|h|\times|h|} \rightarrow \{0,1\}^{|h|\times|h|}$ *are matrix indicator functions.*

Indicator functions are transformations such that function $w_i : \mathbb{R} \rightarrow \{0,1\}$, where $i \in \{1,2,3\}$, is applied to every entry of input matrix independently and identically. We use Equation 10 to define w_i.

$$w_i(x) = \begin{cases} 1 & \text{if } |x| < \hat{x}_i \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

$$\hat{x}_i = \frac{8.72\sqrt{\frac{3}{n}}}{\sum_{j=1}^{N} (max(x_{i,j}) - min(x_{i,j}))^2} \tag{11}$$

Let n be the number of inputs to a single neuron. Let $max(x_{i,j})$ and $min(x_{i,j})$ be the upper bound and the lower bound to the input to neuron j given input data. $\hat{x}_i$ is determined as Equation 11 using ideas from [7]. If a neural network has a high activation score, the parameter assignments are likely to converge to solutions with low error after gradient descent.

Algorithm 2 is a modified version of PSO algorithm that finds the global best particle in each epoch. Because particles are pulled towards the same global best particle, we call this modification as PSO synchronous update. Algorithm 3 for updating activation list AL is called at

Algorithm 2: Synchronous PSO

Data: Number of particles p, Maximum epochs n_1, constant c_1, c_2, Activation list size
Result: A list of weight assignments E
1 $L \leftarrow$Array of Random Particles(p)
2 $AL \leftarrow$New Activation List
3 **for** $i \in (1, .., n_1)$ **do**
4 **for** $j \in (1, .., p)$ **do**
5 $\phi_1, \phi_2 \leftarrow random\_numbers$
6 $L[j].velocity \leftarrow L[j].velocity + \phi_1 c_1 (L[j].l_{best} - L[j].loc) + \phi_2 c_2 (g_{best} - L[j].loc)$
7 $L[j].loc \leftarrow L[j].loc + L[j].velocity$
8 **if** $MSE(L[j].loc) \text{¡} MSE(L[j].l_{best})$ **then**
9 $L[i].l_{best} \leftarrow L[i].loc$
10 **end**
11 **end**
12 $g_{best} \leftarrow min_{q \in L}(MSE(q))$
13 Call Update Activation List(AL)
14 **end**
15 $E \leftarrow AL \cup \{g_{best}\}$
16 Return E

Algorithm 3: Updating Activation List

Data: Size of selection list q, priority queue AL
Result: Updated activation list AL
1 list$\leftarrow$Select q particles with lowest MSEs
2 Sort list by Activation Score in descending order
3 **for** $i \in (1, .., q)$ **do**
4 **if** *AL not full* **then**
5 AL.insert(*list[i]*)
6 **else**
7 **if** $AS(top(AL)) \text{¡} AS(list[i])$ **then**
8 pop(AL)
9 AL.insert(*list[i]*)
10 **else**
11 **if** $i = 1 \wedge MSE(top(AL)) < MSE(list[i])$ **then**
12 pop(AL)
13 AL.insert(*list[i]*)
14 **end**
15 break
16 **end**
17 **end**
18 **end**

line 13 of Algorithm 2. AL is a priority queue that stores weight assignments with high activation scores and low MSEs. In Algorithm 3, the function AS computes the activation score of a particle using Equation 9. q particles with smallest MSEs are inserted into AL. If AL is full, it pops the particle with the largest error if the particle has activation score less than the particle to be inserted. However, a particle with large MSE and large activation score can block a full list. Thus, lines 11-14 of Algorithm 3 is a mutation stage that allows insertion of a particle with lower activation score and smaller MSE.

B. Exploitation Phase

When back propagation algorithm is applied to an unfolded recurrent neural network, hidden layer errors δ_t are computed using Δ_t at the same time stamp. It is possible to aggregate error from all future timestamps via links between context layer and hidden layer. This part of error is $C^T\delta_{t+1}$ according to dynamic programming. Let t_{max} be the total number of timestamps. Let γ_t be a vector of size $|h|$ representing the accumulated error for hidden layer. $\gamma_{t_{max}}$ is set to be equal to $\delta_{t_{max}}$. Equation 13 is proposed to compute hidden layer error instead of using Equation 5 when $t < t_{max}$. The error back propagated from next layer, namely δ_{t+1}, is averaged by the number of hidden nodes using Equation 12. The following theorem shows the new objective errors to be minimized.

$$\gamma_t = (B^T\Delta_t + \frac{1}{|h|}C^T\gamma_{t+1}) \odot f'(Aa_t + Cc_t) \tag{12}$$

$$\delta_t = (B^T\Delta_t + C^T\gamma_{t+1}) \odot f'(Aa_t + Cc_t) \tag{13}$$

Theorem III.2. *The objective errors of hidden layer to be minimized at timestamp t in unfolded Elman neural network defined by δ_t are $\Delta_t + \sum_{i=t+1}^{t_{max}} \frac{\Delta_i}{|h|^{i-t-1}}$.*

Proof. We proceed by induction from $t = t_{max}$ to $t = 0$. Objective errors can be back propagated through weight matrices by chain rule.

Base case: When $t = t_{max}$, proof is trivial by Equation 5.

Inductive step: Assume $\forall\ t \geq \tau$, the objective errors of hidden layer to be minimized at timestamp t defined by δ_t are $\Delta_t + \sum_{i=t+1}^{t_{max}} \frac{\Delta_i}{|h|^{i-t-1}}$. We can decompose $\delta_{\tau-1} = x_{\tau-1} + y_{\tau-1}$, where $x_{\tau-1} = B^T\Delta_{\tau-1} \odot f'(Aa_{\tau-1} + Cc_{\tau-1})$ and $y_{\tau-1} = C^T\gamma_\tau \odot f'(Aa_{\tau-1} + Cc_{\tau-1})$. It follows that $x_{\tau-1}$ is exactly the error back propagated from $\Delta_{\tau-1}$. It suffices to show that the objective error of $y_{\tau-1}$ is $\sum_{i=\tau}^{t_{max}} \frac{\Delta_i}{|h|^{i-t-1}}$. We can decompose $\gamma_\tau = x'_\tau + y'_\tau$ into two parts: $x'_\tau = B^T\Delta_t \odot f'(Aa_\tau + Cc_\tau)$ and $y'_\tau = \frac{1}{|h|}C^T\gamma_{\tau+1} \odot f'(Aa_\tau + Cc_\tau)$. Similarly for $\delta_\tau = x_\tau + y_\tau$, where $x_\tau = B^T\Delta_\tau \odot f'(Aa_\tau + Cc_\tau)$ and $y_\tau = C^T\gamma_{\tau+1} \odot f'(Aa_\tau + Cc_\tau)$. x_τ and x'_τ are identical. They represent the part of objective errors propagated from Δ_τ. We also observe that $y'_\tau = \frac{y_\tau}{|h|}$.

By inductive assumption, objective output errors of δ_τ are $\Delta_\tau + \sum_{i=\tau+1}^{t_{max}} \frac{\Delta_i}{|h|^{i-\tau-1}}$. Hence y_τ has objective error $\sum_{i=\tau+1}^{t_{max}} \frac{\Delta_i}{|h|^{i-\tau-1}}$, which implies y'_τ has objective errors $\frac{1}{|h|}\sum_{i=\tau+1}^{t_{max}} \frac{\Delta_i}{|h|^{i-\tau-1}}$. Thus, γ_τ is the error for $\Delta_\tau + \frac{1}{|h|}\sum_{i=\tau+1}^{t_{max}} \frac{\Delta_i}{|h|^{i-\tau-1}} = \Delta_\tau + \sum_{i=\tau}^{t_{max}} \frac{\Delta_i}{|h|^{i-\tau-1}}$. Since the objective error of $y_{\tau-1}$ is identical of γ_τ by chain rule, it follows that the inductive assumption holds for $t = \tau - 1$. □

A direct implication of this theorem is that the objective error function to be minimized at any time stamp contains all errors in its future. Moreover, the weights of errors decay in terms of time. Unlike traditional back propagation error definition, the output errors in later timestamps are repeatedly used at the gradient computation before gradient averaging phase in our proposed method. Experimental results have shown the the proposed error computation can improve both convergence speed and accuracy for time series data.

C. Complexity Analysis

The complexity for computing all outputs of an recurrent neural network is $O(t_{max}(|a||h| + |o||h| + |h|^2))$, so Algorithm 2 has complexity $O(n_1pt_{max}(|a||h| + |o||h| + |h|^2))$ excluding line 13 given p particles and n_1 epochs as input. Algorithm 3 can reuse the MSEs computed in the same epoch. The rest part does not depend on t_{max}. Thus, the overall complexity for exploration phase is $O(n_1pt_{max}(|a||h| + |o||h| + |h|^2))$.

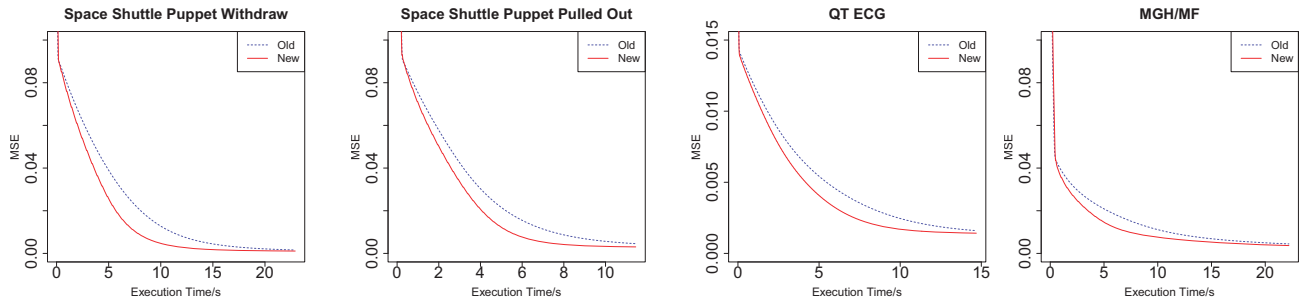
The complexity for exploitation can be divided into two parts: error back propagation and weights update. For error back propagation, the unfolded network has t_{max} number of layers. Computation for errors has complexity $O(t_{max}|h|(|h| + |a| + |o|))$. Gradient computation based on errors has the same complexity. Weights update operates on folded recurrent neural network, so this process has complexity of $O(|a||h| + |o||h| + |h|^2)$. Suppose AL has size m, the entire exploitation phase has complexity $O(n_2mt_{max}(|h|^2 + |a||h| + |o||h|))$, given n_2 epochs.

$$\frac{n_1t_0 + n_2mt_1}{mkt_2} = \frac{n_1p}{mk}\beta + \frac{n_2}{k}\alpha \tag{14}$$

Let t_0 and t_1 be the execution time of one epoch in synchronous PSO and enhanced back propagation respectively. Equation 14 shows the ratio of hybrid algorithm to traditional back propagation for m number of recurrent neural networks with k number of epochs. α and β are constants. We can compare the end-to-end execution time of proposed hybrid algorithm and traditional back propagation using Equation 14.

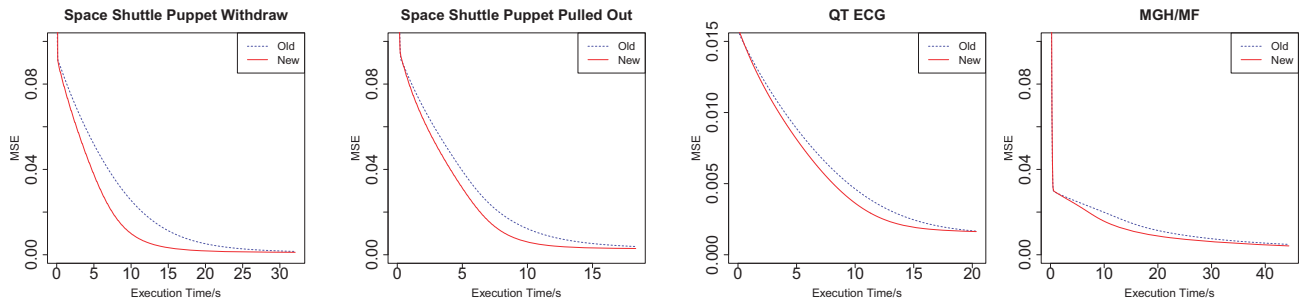
IV. Experimental Results

We use four datasets to evaluate our algorithm: the QT ECG dataset[8], two NASA space shuttle datasets[10], and



a b c d

Figure 3: MSE of testing data against epoch under rBP. (a) Space shuttle puppet withdraw (b) Space shuttle puppet pulled out. (c) QT ECG database record 0606. (d) MGH/MF database record 0003.



a b c d

Figure 4: MSE of testing data against epoch under ADAM. (a) Space shuttle puppet withdraw (b) Space shuttle puppet pulled out. (c) QT ECG database record 0606. (d) MGH/MF database record 0003.

Table I: Datasets characteristics.

Dataset	Timestamps	Dimension
QT ECG	1000	1
Space Shuttle Puppet Withdraw	4000	1
Space Shuttle Puppet Pulled out	2000	1
MGH/MF	3600	3

MGH/MF dataset[9] recording three-dimensional heart pressures. Table I summarizes all datasets.

QT ECG dataset measures the electrical voltage of human heart beats at different timestamps[8]. NASA space shuttle datasets illustrate the sensor records of energizing phases of space shuttle Marotta Valves. MGH/MF dataset[9] contains multiple electrical recordings of patients' conditions. For MGH/MF dataset, we focus on three attributes: arterial pressure, pulmonary arterial pressure, central venous pressure in particular because it has been argued in [19] that the heart pressures are correlated, which enables the use of AR model.

A. Enhanced Recurrent Back Propagation

Current research works focus on gradient descent phase of back propagation algorithm in order to improve convergence speed and accuracy. Our proposed enhanced back propagation changes the computation of objective error in the error back propagation stage, so it can be combined with existing gradient descent methods. In this section, we demonstrate the convergence speed and final MSE improvement for two state-of-art methods: l_2-regularization back propagation (rBP) and adaptive moment estimation-based method (ADAM).

$$\frac{1}{m}\sum_{i=1}^{m}(old[i] - new[\alpha \times i])/old[i] \tag{15}$$

Equation 15 defines discrete average MSE convergence speedup. It presents that the average percentage of error reduction given the same execution time. Given a recurrent neural network, $old[i]$ refers to the MSE of the traditional back propagation and $new[i]$ refers to the MSE of enhanced back propagation for running i number of epochs. Because the proposed error aggregation bears the overhead of extra matrix-vector multiplications per epoch, the comparison of *old* and *new* should have a phase shift in order to compute average MSE convergence speedup given the same execution time. The phase shift percentage is represented by α in Equation 14 and 15.

We initialize 32 recurrent neural networks with randomized weights. For each of the networks, we apply the same gradient descent algorithm using errors computed by Equations 5 and 13 separately to the same training data with the same number of epochs. Learning rate ranging from 0.5 to 1.0 and $\lambda = 0.01$ are used for rBP. We evaluate convergence speedup and error reduction at convergence using testing data with two-fold cross validation and average results of 32 networks.

Figures 3a, 3b, 3c, 4a, 4b, and 4c illustrate testing error convergence in terms of execution time. Table II and

Table II: The improvements from new error definition for rBP gradient descent.

Dataset	MSE Convergence Speedup	MSE reduction
SSPW	18.1%	9.38%
SSPO	16.8%	6.83%
QT-ECG	10.2%	13.3%
MGH/MF	17.2%	28.3%

Table III: The improvements from new error definition for ADAM gradient descent.

Dataset	MSE Convergence Speedup	MSE reduction
SSPW	24.0%	17.4%
SSPO	23.6%	11.8%
QT-ECG	9.74%	25.7%
MGH/MF	15.2%	27.5%

Table III summarizes the average convergence speedup and MSE percentage reduction at convergence for all datasets. Although Equation 13 results longer execution time per epoch, the gradient descent algorithm converges faster given the same execution time. Moreover, the new error definition gives lower convergence errors. In addition, as we increase the size of dataset, both average MSE convergence speedup and MSE percentage reduction at convergence are improved.

B. Hybrid Algorithm

With the help of exploration phase, the hybrid algorithm can further reduce convergence error. $n_1 = 10$ is set to be the number of epochs for exploration phase. $m = 32$ recurrent neural networks with randomized weights are trained using both ADAM and rBP using traditional back propagation as benchmarks. To make a fair comparison, the activation list size is set to be 31, so $m = 32$ recurrent neural networks are produced by exploration phase and trained with enhanced back propagation algorithm.

Table IV summarizes the results of two-fold cross validation for improvement of MSE using the hybrid algorithm. MSE reduction column shows the percentage MSE reductions achieved by the hybrid method when different gradient descent methods are used. The average MSE reductions are higher compared with the results shown in Table II for all datasets. However, the execution time of the hybrid algorithm is approximately 40% to 50% longer. According to Equation 14, if n_2 and m become large, the execution time ratio of old method to new method approach to α given fixed n_1. Moreover, decreasing either of n_1 and n_2 reduces execution time, but resulting larger error. Thus, there is a trade off between execution time and accuracy, users should tune the values of n_1 and n_2 depending on the timing and accuracy constraints.

Acknowledgment

This work is supported in part by the following grants: NSF award CCF-1409601; DOE awards DE-SC0007456, DE-SC0014330; AFOSR award FA9550-12-1-0458; NIST award 70NANB14H012.

Table IV: MSE percentage reductions of hybrid algorithm (including all phases) for rBP and ADAM gradient descent.

Dataset	rBP MSE Reduction	ADAM MSE Reduction	Percentage Time Increase
SSPW	29.0%	38.6%	52%
SSPO	28.9%	35.1%	53%
QT-ECG	64.5%	66.4%	41%
MGH/MF	45.6%	44.8%	50%

References

[1] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[2] A. Hajdarevic, L. Banjanovic-Mehmedovic, I. Dzananovic, F. Mehmedovic, and M. A. Ahmad, "Recurent neural network as a tool for parameter anomaly detection in thermal power plant," *International Journal of Scientific and Engineering Research (IJSER)*, vol. 6, no. 8, pp. 448–455, 2015.

[3] P. Xiao, G. K. Venayagamoorthy, and K. A. Corzine, "Combined training of recurrent neural networks with particle swarm optimization and backpropagation algorithms for impedance identification," in *2007 IEEE Swarm Intelligence Symposium*, pp. 9–15, IEEE, 2007.

[4] B. Srinivasan, U. R. Prasad, and N. J. Rao, "Back propagation through adjoints for the identification of nonlinear dynamic systems using recurrent neural models," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 213–228, 1994.

[5] B. Poole, J. Sohl-Dickstein, and S. Ganguli, "Analyzing noise in autoencoders and deep networks," *arXiv preprint arXiv:1406.1831*, 2014.

[6] Y. Shi *et al.*, "Particle swarm optimization: developments, applications and resources," in *evolutionary computation, 2001. Proceedings of the 2001 Congress on*, vol. 1, pp. 81–86, IEEE, 2001.

[7] J. Y. Yam and T. W. Chow, "Feedforward networks training speed enhancement by optimal initialization of the synaptic coefficients," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 430–434, 2001.

[8] P. Laguna, R. G. Mark, A. Goldberg, and G. B. Moody, "A database for evaluation of algorithms for measurement of qt and other waveform intervals in the ecg," in *Computers in Cardiology 1997*, pp. 673–676, IEEE, 1997.

[9] J. Welch, P. Ford, R. Teplick, and R. Rubsamen, "The massachusetts general hospital-marquette foundation hemodynamic and electrocardiographic database–comprehensive collection of critical care waveforms," *Clinical Monitoring*, vol. 7, no. 1, pp. 96–97, 1991.

[10] B. Ferrell and S. Santuro, "Nasa shuttle valve data," 2005.

[11] H. Hyötyniemi, "Turing machines are recurrent neural networks," *Proceedings of step*, vol. 96, 1996.

[12] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, no. 34, p. 13, 2001.

[13] M. I. Jordan, "Serial order: A parallel distributed processing approach," *Advances in psychology*, vol. 121, pp. 471–495, 1997.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] X. Gao, X. Gao, and S. Ovaska, "A modified elman neural network model with application to dynamical systems identification," in *Systems, Man, and Cybernetics, 1996., IEEE International Conference on*, vol. 2, pp. 1376–1381, IEEE, 1996.

[16] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] P. Shelokar, P. Siarry, V. K. Jayaraman, and B. D. Kulkarni, "Particle swarm and ant colony algorithms hybridized for improved continuous optimization," *Applied mathematics and computation*, vol. 188, no. 1, pp. 129–142, 2007.

[18] J.-R. Zhang, J. Zhang, T.-M. Lok, and M. R. Lyu, "A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training," *Applied mathematics and computation*, vol. 185, no. 2, pp. 1026–1037, 2007.

[19] D. T. Mangano, "Monitoring pulmonary arterial pressure in coronary-artery disease.," *Anesthesiology*, vol. 53, no. 5, pp. 364–370, 1980.