Mitigation of Catastrophic Forgetting in Recurrent Neural Networks using a Fixed Expansion Layer

Robert Coop, *student member*, *IEEE*, and Itamar Arel, *senior member*, *IEEE*

***Abstract*—Catastrophic forgetting (or catastrophic interference) in supervised learning systems is the drastic loss of previously stored information caused by the learning of new information. While substantial work has been published on addressing catastrophic forgetting in memoryless supervised learning systems (e.g. feedforward neural networks), the problem has received limited attention in the context of dynamic systems, particularly recurrent neural networks. In this paper, we introduce a solution for mitigating catastrophic forgetting in RNNs based on enhancing the Fixed Expansion Layer (FEL) neural network which exploits sparse coding of hidden neuron activations. Simulation results on several non-stationary data sets clearly demonstrate the effectiveness of the proposed architecture.**

I. Introduction

Catastrophic interference (known also as catastrophic forgetting) is a well known characteristic of many parametric learning systems where the exposure of the learning system to new information results in severe loss of previously learned information. This extreme loss of information cannot be attributed to the inherent informational capacity of the learning system (as determined by the amount of resources available for learning), but rather is caused by the overlap of representations of information within the learning system [7], [8].

Many real-world applications, such as financial time series analysis and climate prediction, involve data streams that are either strictly non-stationary or can at best be considered piecewise stationary. The situation can be further exacerbated by time and resource constraints which rule out batch processing and instead necessitate online, sequential learning. These problem domains require algorithms which address the stability-plasticity dilemma [11] and are able to find a balance between the capacity to learn new information and the ability to retain previously learned representations. In this sense, the problem of catastrophic interference has much in common with the task of learning in nonstationary environments or learning under concept drift [4].

Recent studies have suggested that catastrophic interference can also be observed in mammals. In experimental studies, it has been demonstrated that long durations of time between observations of stationary patterns can lead to an excessive tendency to form associations between sensory inputs and desired outputs (abnormal potentiation) at the expense of weakening existing associations [15], [10].

Robert Coop and Itamar Arel are with the Min-Kao Department of Electrical Engineering & Computer Science, The University of Tennessee, Knoxville, Tennessee (email: {coop,itamar}@eecs.utk.edu).

Addressing this challenge has been the focus of numerous studies in recent years. However, the majority of these studies focused their attention on memoryless, or static, systems rather than dynamical models. A classic example of static models are multilayer perceptrons, which can be viewed as parameterized models for mapping the input space to an output space. In contrast, recurrent neural networks employ a feedback signal to introduce memory in modeling dynamical systems. This paper addresses the issue of mitigating catastrophic forgetting in recurrent neural networks by expanding on prior work which was devised for feedforward architectures [5].

The rest of this paper is structured as follows. Section II reviews some of the existing schemes published in the literature for mitigating catastrophic forgetting. Section III discusses the Fixed Expansion Layer (FEL) neural network, which was recently proposed as means of addressing the forgetting effects in feedforward architectures [3], and describes the enhancement of the FEL network in the context of an RNN. In Section IV, simulation results are presented while in Section V the conclusions are drawn.

II. Background

Notably, most mainstream exploration of the problem of catastrophic interference has been within the domain of auto-associative pattern learning, which has not specifically addressed problems inherent with classification or more general function approximation [16]. The following outlines the most commonly utilized schemes proposed in the literature.

A. Alleviating Catastrophic Interference

Traditional approaches to the problem can be divided into three different categories based on their operation: methods that involve the rehearsal of patterns or pseudopatterns, methods which use dual networks in order to store learned information, and methods based on modifying the network's operation.

Rehearsal methods involve augmenting the actual training data with saved (or artificially generated) data during training. Standard rehearsal methods set samples of data aside during training and later reuse these samples in order to prevent loss of information; sweep rehearsal [18] and the rehearsal buffer model [17] are two examples of these. Pseudorehearsal uses patterns generated from the network itself (known as pseudopatterns) during the rehearsal process (e.g. [19], [1], [9]). Random data is fed into the network, and the resulting output is stored. Later, the same data is fed into the network with the earlier recorded response being used

as a training target. These pseudopatterns serve as indirect snapshots of the network's internal state at some time during the training process, with later rehearsal acting to re-adjust the network in order to be more like the prior snapshot. This adjustment causes the network to be more likely to retain prior information, thus combating the catastrophic interference effects.

Dual network methods use two neural networks in order to separate the task of learning from the task of long-term information storage (e.g. [12], [1], [7]). These methods use one network (the 'learning' network) for the training process and then transfer information to a second network (the 'memory' network). Pseudopatterns are used in order to transfer information from one network to another; during training the memory network is trained with pseudopatterns generated from the learning network in order to store information, and during subsequent training the learning network is trained with pseudopatterns generated from the memory network in order to refresh previous information and prevent data loss.

Activation sharpening, a method based on the modification of a network's operation, is inspired by the belief that catastrophic forgetting is a consequence of the overlap of pattern representations within the neural network and can be addressed by reducing such overlap [7]. The goal of activation sharpening is to gradually develop semi-distributed representations of patterns in the hidden layer of the network by causing neurons within the hidden layer to 'latch' onto specific regions of the input space. This approach modifies the traditional feedforward process; the input pattern is fed forward and the activation of neurons in the hidden layer is 'sharpened' by increasing the activation of one or more of the hidden neurons (selected by choosing the hidden neurons with the largest activation values) and decreasing the activation values of all the other hidden neurons. The difference between the original and sharpened activation values is immediately backpropagated to the input-hidden weights (as if it was an error signal) in order to train the network to produce a sharpened activation in the future.

B. Recurrent Learning of Temporal Sequences

The standard multilayer perceptron neural network does not have the ability to approximate temporally-dependent mappings between its input space and its output space. To address this, Elman's Simple Recurrent Network (SRN) [6] utilizes 'context' neurons which form a representation of the current state, as depicted in Figure 1. Following each feedforward process, the values of the hidden neurons are stored into the context neurons. These context neurons are fed to the network, in addition to the external inputs, during the subsequent time step. Using these memory signals the network is able to capture temporal dependencies. SRNs, and most gradient-based recurrent neural network models, have difficulty with capturing long sequences, or functions with long-term temporal dependencies, due to the 'vanishing gradient' phenomenon [2]. Regardless, the non-stationary nature of real-world data sequences typically fed to recurrent networks renders the task of accurately modeling processes difficult.

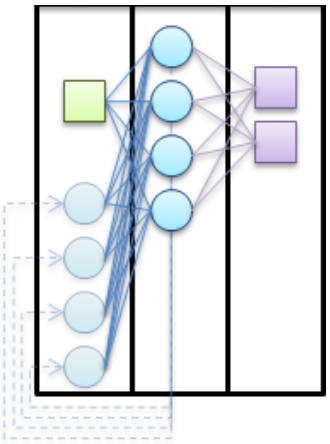

Figure 1. The Simple Recurrent Network (SRN) architecture [6]. The activation values of the hidden layer neurons at time (t) are treated as part of the input vector at time $(t+1)$.

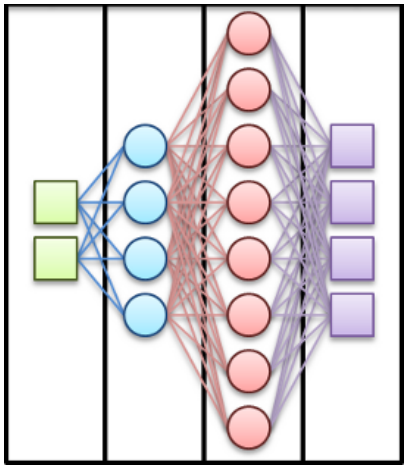

Figure 2. Fixed Expansion Layer Feedforward Neural Network

III. The Fixed Expansion Layer Network

The motivation behind the fixed expansion layer (FEL) neural network, which was first introduced by the authors in the context of feedforward neural networks [3], is similar to the motivation for activation sharpening: reducing the overlap of pattern representations within the network. The FEL network addresses the problem of representational overlap by exploiting an augmented MLP architecture which includes the addition of an *expansion* hidden layer to the network, as depicted in Figure 2. The weights for this layer are fixed during network initialization and remain unchanged during subsequent network operations. As a result, the FEL framework inherently supports an incremental, online learning process and exploits sparse encoding to latch onto previously learned input/output mappings.

During the feedforward phase, the FEL neurons are triggered in order to present a consistent sparse representation of the input pattern to the output layer, as illustrated in Figure

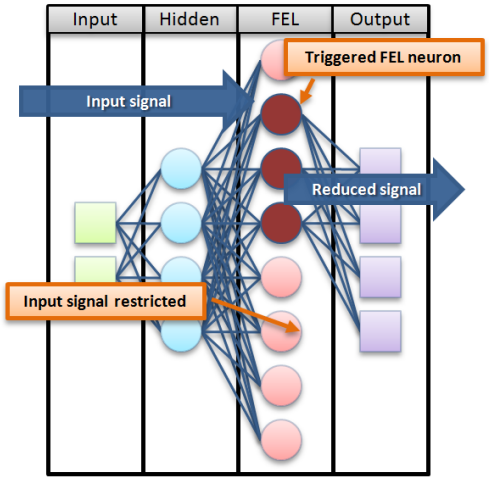

Figure 3. Feedforward signal flow in the FEL neural network

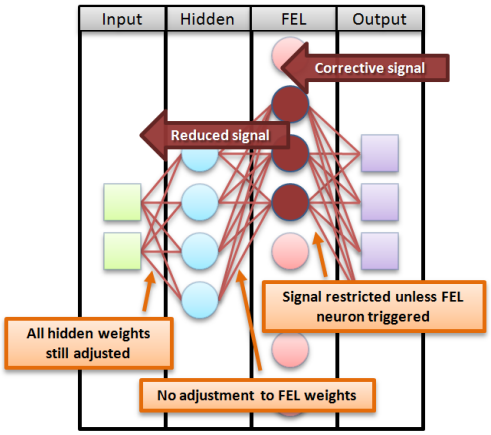

Figure 4. Error back-propagation in the FEL neural network

3. The sparsity of the triggered FEL neurons protects the input-to-hidden layer weights from portions of the back-propagated error signal (see Figure 4), thus preventing the network weights from changing drastically when exposed to new information, which mitigates the effects of catastrophic interference. Sparsity thus serves as means of latching on to older information by selectively gating weight update signal propagation through the network.

Consistency of representations within the FEL neurons is critical to the accuracy of the network; the network cannot retain information if the dense signal from the hidden layer is not sparsely encoded in a consistent fashion. This requirement is the main factor to consider in deciding how the activation values of the FEL neurons are to be calculated.

One approach to the FEL activation problem is to perform a traditional feedforward operation and then to sparsify the FEL activation values. Using the fixed weights and the activation values of the hidden layer, activation values are calculated for all FEL neurons. Once these values have been

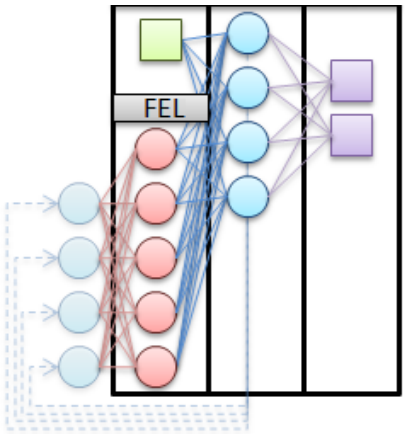

Figure 5. The Recurrent Fixed Expansion Layer (rFEL) Network architecture. The activation values of the hidden layer neurons at time t are sparsely encoded using the fixed weights, and the sparse encoding present in the fixed expansion layer neurons is treated as part of the input vector at time $t+1$.

calculated, we heuristically determine which FEL neurons contain the most significant information about the hidden layer and zero out the values of the other FEL neurons. This approach was investigated in [3], but has a number of drawbacks. Most significantly, heuristically determining the FEL activation in this fashion requires a number of constant values to be determined and can lead to significant parameter tuning for each problem domain.

An improved approach is to ensure sparsity by minimizing the L_1 norm of the fixed expansion layer neurons, while at the same time retaining key information from the hidden layer neurons. This provides a structured approach for calculation of FEL activation values and ensures consistency of representations within the FEL neurons. Moreover, such an approach is equivalent to solving the following optimization problem formulation:

$$\min_x ||y - Ax||^2 + \gamma||x||_1, \tag{1}$$

where y denotes the activation of the hidden layer, A is the FEL weight matrix, x is the FEL activation (over which the minimization is performed), and γ is a penalty constant which acts to balance the desire for a sparse activation signal and retention of information from the hidden layer.

This problem can be efficiently solved using the feature-sign search algorithm introduced in [14]. This search algorithm hinges on the fact that equation (1) can equivalently be written as

$$\min_x ||y - Ax||^2 + \gamma \sum_{i=1}^{N} |x_i|, \tag{2}$$

where N is the number of elements in x, and if we know the signs of the elements in x then we can replace each of the terms $|x_i|$ with either x_i (if $x_i > 0$), $-x_i$ (if $x_i < 0$), or 0 (if $x_i = 0$). Considering only nonzero coefficients, this reduces (2) to a standard, unconstrained quadratic optimization problem, which can be solved analytically and efficiently. The feature-sign search algorithm operates by

Feature-sign search algorithm

1: Initialize $x := \vec{0}$, $\theta := \vec{0}$, and $active\ set := \{\}$, where $\theta_i \in \{-1, 0, 1\}$ denotes sign(x_i).

2: From zero coefficients of x, select $i = \arg\max_i \left|\frac{\partial ||y - Ax||^2}{\partial x_i}\right|$.

Activate x_i (add i to the $active\ set$) only if it locally improves the objective, namely:

If $\left|\frac{\partial ||y - Ax||^2}{\partial x_i}\right| > \gamma$, then set $\theta_i := -1$, $active\ set := \{i\} \cup active\ set$.

If $\left|\frac{\partial ||y - Ax||^2}{\partial x_i}\right| < -\gamma$, then set $\theta_i := 1$, $active\ set := \{i\} \cup active\ set$.

3: Feature-sign step:

Let $\hat{A}$ be a submatrix of A that contains only the columns corresponding to the $active\ set$.

Let $\hat{x}$ and $\hat{\theta}$ be subvectors of x and θ corresponding to the $active\ set$.

Compute the analytical solution to the resulting unconstrained QP (minimize$_{\hat{x}} ||y - \hat{A}\hat{x}||^2 + \gamma\hat{\theta^T}$):

$\hat{x}_{new} := (\hat{A}^T\hat{A})^{-1}(\hat{A}^T y - \gamma\hat{\theta}/2)$,

Perform a discrete line search on the closed line segment from $\hat{x}$ to $\hat{x}_{new}$:

Check the objective value at $\hat{x}_{new}$ and all points where any coefficient changes sign.

Update $\hat{x}$ (and the corresponding entries in x) to the point with the lowest objective value.

Remove zero coefficients of $\hat{x}$ from the $active\ set$ and update $\theta := \text{sign}(x)$.

4: Check the optimality conditions:

(a) Optimality condition for nonzero coefficients: $\frac{\partial ||y - Ax||^2}{\partial x_i} + \gamma\text{sign}(x_j) = 0,\ \forall x_j \neq 0$

If condition (a) is not satisfied, go to Step 3 (without any new activation); else check condition (b).

(b) Optimality condition for zero coefficients: $\frac{\partial ||y - Ax||^2}{\partial x_i} \leq \gamma,\ \forall x_j = 0$

If condition (b) is not satisfied, go to Step 2; otherwise return x as the solution.

Table I
THE FEATURE-SIGN SEARCH ALGORITHM [14]

maintaining an *active set* of potentially non-zero coefficients in x and the sign (positive or negative) of these values; all other members of x are assumed to be zero. Given an active set, we can analytically solve (2); in addition, given the corresponding minimization solution, an improved active set can be obtained (if it exists).

Each such step reduces the objective function. It has been shown that the feature-sign search algorithm converges to a global optimum of the optimization problem (2) in a finite number of steps [14]. Previous work (submitted for publication) has shown that this improved FEL activation method results in greatly increased accuracy. The algorithm is shown in Table I.

A. The Recurrent FEL Network

The standard multilayer perceptron neural network does not have the ability to approximate temporally-dependent mappings between its input space and its output space. To address this, Elman's Simple Recurrent Network (SRN) [6] utilizes 'context' neurons which form a representation of the current state. Following each feedforward process, the values of the hidden neurons are stored into the context neurons (as depicted in Figure 1). These context neurons are fed to the network, in addition to the external inputs, during the subsequent time step.

Formally, consider a multilayer perceptron which has N input neurons $[x_1, x_2, \ldots, x_N]$ and M hidden neurons $[h_1, h_2, \ldots, h_M]$. To transform this MLP into a recurrent network, context neurons are added to the input layer. The M new context neurons $[x'_1, x'_2, \ldots, x'_M]$ will store the prior value of the hidden layer neurons. The hidden layer calculation for this recurrent network is given by

$$h_j^{(t)} = f\left(\sum_{i=1}^{N} w_{ji}x_i + \sum_{i=1}^{M} w'_{ji}x'_i\right) \tag{3}$$

where $h_j^{(t)}$ is the value of the j-th hidden neuron at time t and w'_{ji} is the weight that the j-th hidden neuron assigns to the i-th context neuron. The values of the context neurons x'_i are given by

$$x'_j = h_j^{(t-1)} \qquad j \in [1, M] \tag{4}$$

where $h_j^{(t-1)}$ is the value of the j-th hidden neuron from the previous input at time $t-1$.

Applying the FEL technique to the SRN allows the system to capture temporal dependencies. In this recurrent FEL (rFEL) network, the expansion layer now acts to prevent convergence of the weights between the context neurons and the hidden layer neurons. This is done by gating the context signal using an expansion layer, as depicted in Figure 5.

Specifically, we define a new context signal calculated by using the feature-sign search algorithm to solve

$$\min_{c^{(t)}} ||h^{(t-1)} - Ac^{(t)}||^2 + \gamma||c^{(t)}||_1 \tag{5}$$

where $c^{(t)}$ is the context signal at time t, A is a matrix of the fixed expansion weights, $h^{(t-1)}$ is the vector of hidden neuron activations for time $(t-1)$, and γ is a constant determining the relative importance between the sparsity of $c^{(t)}$ and the fidelity of the signal reconstruction given by $Ac^{(t)}$.

Using this new context signal, Equation 3 becomes

$$h_j^{(t)} = f\left(\sum_{i=1}^{N} w_{ji}x_i + \sum_{i=1}^{O} w'_{ji}c_i^{(t)}\right) \tag{6}$$

where O is the number of context neurons (i.e. the number of dimensions of the vector $c^{(t)}$ as previously defined), $c_i^{(t)}$ is the i-th context neuron's value, and w'_{ji} is the weight that the j-th hidden neuron assigns to the i-th context neuron.

IV. Simulation Results

We performed several simulations in order to evaluate the proposed recurrent FEL (rFEL) network's ability to mitigate catastrophic interference. Performance in each case is compared to several other algorithms proposed in the literature.

A. Auto-associative binary sequence reconstruction

Auto-associative binary sequence reconstruction can be viewed as a sequence-based version of a traditional catastrophic forgetting task. Mainstream exploration of the problem of catastrophic interference has commonly been restricted to the domain of auto-associative pattern learning [16]; this task compares the ability of a network to retain previously captured sequences of observations after demonstrating the learning of new sequences. In particular, the capacity of networks to retain information about sequences of patterns is evaluated, rather than that of a single presentation for each pattern.

The auto-associative binary sequence reconstruction test was performed as follows. A number of binary pattern sequences were randomly generated, where each sequence consisted of two binary patterns (composed of the values -1 and 1). The objective is to learn to reconstruct the first part of the sequence *after* the complete sequence is observed. For example, given the binary patterns A_1 and A_2, one training iteration would consist of A_1 being input in to the network (with no training target being provided and no backpropagation taking place), then A_2 would be input in to the network (along with the context signal from the previous presentation of A_1). After the presentation of A_2, the goal for the network would be to output the value of A_1; A_1 would be provided as a training target and used in error backpropagation only after the presentation of A_2.

This test was repeated for sequences using patterns consisting of 16, 32, and 64 dimensions in order to measure each algorithm's ability to scale to more challenging settings. First, the network is trained over a batch of 20 randomly generated sequences. The network is repeatedly trained over this batch until the mean squared error for the training set reaches an acceptable level. For the 16-dimensional and 32-dimensional tests, a threshold of 0.06 was used; for the 64-dimensional test, this threshold was increased to 0.075 due to the fact that not all networks were able to reach the 0.06 error level. Once the network has learned the 20 base sequences, a new sequence is presented to the network. The network is trained using this intervening sequence, and then the mean squared error over the original set of base items is measured in order to determine how much of the original information was retained.

This experiment was conducted using the rFEL network, a network using pseudorehearsal, a network using activation sharpening, a system using dual networks, and a standard Simple Recurrent Network (SRN). For the pseudorehearsal, activation sharpening, and dual networks we use SRNs as the base network. All networks used 16 hidden neurons, a

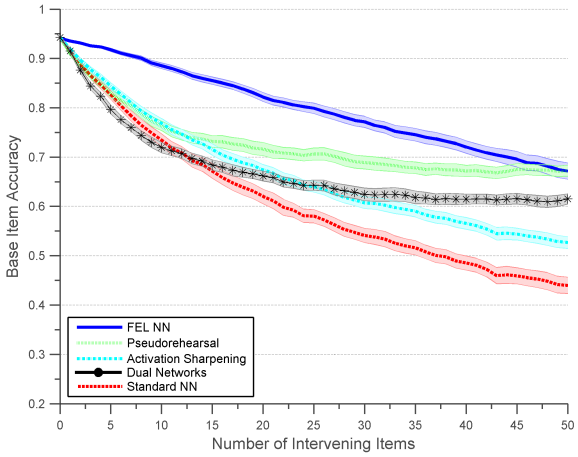

Figure 6. Binary sequence reconstruction: 16-dimensional pattern

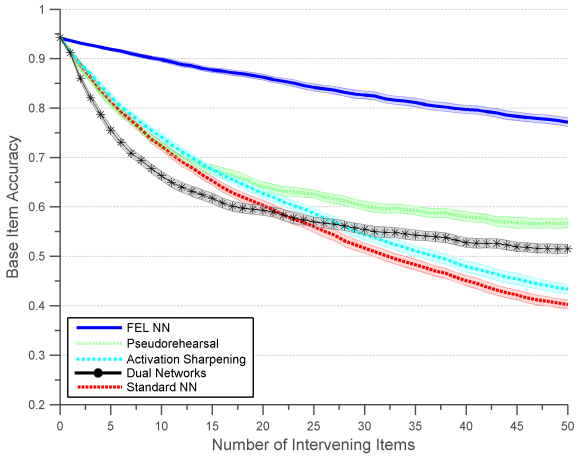

Figure 7. Binary sequence reconstruction: 32-dimensional pattern

step size of 2^{-5}, and were trained with standard stochastic gradient descent. After the pseudorehearsal network was successfully trained, 32 pseudopatterns were generated; a pseudopattern was used for training after the introduction of each new sequence. During training of the dual network system, 20 pseudopatterns were created with the learning network and used to transfer information to the memory network. Once a new sequence was introduced, the learning network was trained using 3 pseudopatterns generated by the memory network. For activation sharpening, 8 neurons were sharpened with a sharpening factor of 0.1, and the rFEL network used a sparse expansion layer with 128 neurons and $\gamma = 2.5$.

For each setting, 50 independent runs were conducted from which the mean accuracy, standard deviation, and the 95% confidence interval for the mean accuracy were derived. These values are shown in Figures 6, 7, and 8; detailed results for the 32-dimensional test are illustrated in Table II. As can be observed from these graphs, a pseudo-linear degradation is exhibited by the recurrent FEL network, while performance

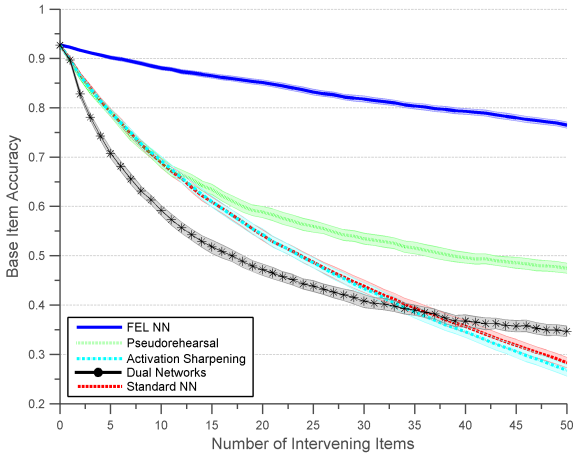

Figure 8. Binary sequence reconstruction: 64-dimensional pattern

# of New Items	Simple Recurrent Network	Pseudo-rehearsal	Activation Sharpening	Dual Networks	**Recurrent FEL**
10	0.7228	0.7253	0.7411	0.6626	**0.8981**
20	0.6031	0.6402	0.6267	0.5929	**0.8623**
30	0.5166	0.6016	0.5440	0.5542	**0.8264**
40	0.4507	0.5805	0.4791	0.5276	**0.7971**
50	0.4024	0.5675	0.4338	0.5151	**0.7714**

Table II
BINARY SEQUENCE RECONSTRUCTION: 32-DIMENSIONAL PATTERN DETAIL. ENTRIES DENOTE THE MEAN ACCURACY OF THE NETWORK OVER 50 INDEPENDENT TEST RUNS.

of other schemes follows a sharper degradation profile.

B. Non-stationary MNIST sequence classification task

The previous experiment demonstrated the viability of the recurrent FEL network within the domain of auto-associative memory of temporal sequences. The next test case was devised in order to evaluate the ability of the rFEL network to classify sequences of complex visual patterns, namely involving the classification of images taken from the MNIST database of handwritten digits [13]. The digits 1, 2, 3, and 4 were selected for this task.

The goal of this task is to correctly classify sequences consisting of two of the four digits. Consequently, four different sequences were considered: $(1,3)$, $(1,4)$, $(2,3)$, $(2,4)$. A training iteration would consist of the presentation of the digit 1, followed by a presentation of the digit 3 (along with the context signal generated during the presentation of the digit 1). After the digit 3 has been presented, the network's goal would be to classify this sequence; there are 4 outputs of the network, each corresponding to one class. The classification decision is made by selecting the output with the maximum value. For training purposes, a target vector is presented which consists of a 1 for the correct class and -1 for all of the other classes (e.g. for the previous example the training target would be $\begin{bmatrix} 1 & -1 & -1 & -1 \end{bmatrix}$.

The original MNIST digits are 32x32 pixel grayscale images, with each image consisting of 1024 pixels taking on integer values from 0 (white) to 255 (black). The MNIST data was preprocessed by shifting the pixel values to be a real number between 0 and 1, centering the data, and using principal component analysis (PCA) to reduce the 1024-dimensional data to 128 dimensions. This dimensionality reduction captured approximately 94% of the variance of the original data. By taking the Moore-Penrose pseudoinverse of the principal component matrix, we can reconstruct the reduced data in order to get a visual representation of the amount of information present in the reduced dataset. Samples of the reconstructed sequences are depicted in Figure 9.

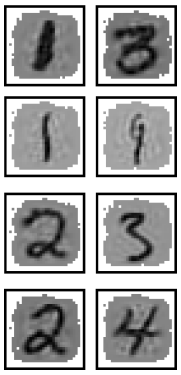

Figure 9. Sample MNIST sequences. Each vertical row is a sample of a sequence used in the classification task, with the two digits being presented sequentially. For example, one training sequence might consist of the presentation of the digit 1 followed by the digit 3, which would be a different class than the digit 1 followed by the digit 4.

When the sample distribution does not change during the training period, all algorithms evaluated were able to achieve over 90% accuracy. In order to employ the MNIST data to study the mitigation of catastrophic interference, the training sequences were presented such that the networks were exposed to nonstationary inputs.

In order to study each algorithm's ability to mitigate catastrophic interference in the presence of non-stationarity, the sample distribution was varied over the training period such that samples were drawn from all classes of sequences during the initial phase of the training process, followed by a duration of time in which samples were drawn only from two of the classes (i.e. 'primary' classes). During this latter phase of the training, no samples from the other two classes (i.e. 'restricted' classes) were presented. Testing was performed over both the primary and the restricted classes with the goal being to determine the degree to which the network was capable of retaining information about the restricted classes after being presented with multiple observations drawn only from the primary classes. The proportion of training iterations that pertained to the primary classes (i.e. the 'non-stationary percentage') was varied in order to measure how well each algorithm performed under different levels of interference.

For this task, a total of 8,000 training samples were presented. During the first phase of the training process, samples were drawn randomly from all classes with equal

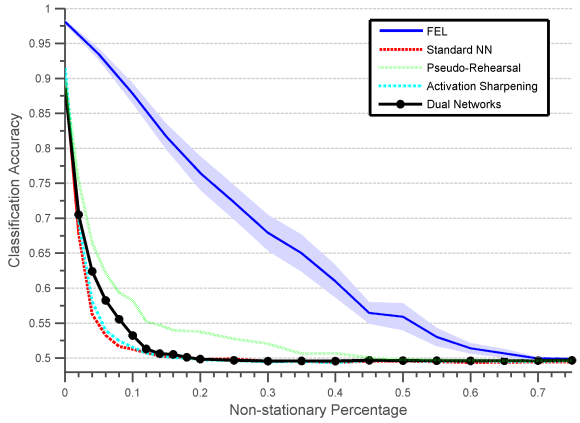

Figure 10. MNIST sequence classification accuracy

Figure 11. Recurrent FEL neuron affinity

probability, while at the second phase of training, samples were drawn only from the two primary classes. The restricted sequences were $(1,3)$, $(1,4)$, and the primary sequences were $(2,3)$, $(2,4)$.

The same algorithms used in the previous section were applied in this experiment. All networks hosted 64 hidden neurons, used a step size of 2^{-5} and were trained using standard stochastic gradient descent. For the pseudorehearsal algorithm, pseudopatterns were created and occasionally used during the training process; following every 400 training iterations, 50 new pseudopatterns were generated and added to the collection of pseudopatterns, and 25 random pseudopatterns were drawn from the collection and used for training. The dual network system was trained in a similar fashion. Following every 400 training samples, 50 pseudopatterns were created and used to transfer information into the memory network, then 100 pseudopatterns were created and used to transfer information to the learning network. For activation sharpening, 32 neurons were sharpened with a sharpening factor of 0.001, and the rFEL network used a sparse expansion layer with 512 neurons and $\gamma = 0.4$; the lower γ value reflects the need for more contextual information than in the previous task.

Each network was used in 35 independent runs; the results were averaged and are shown in Figure 10. The 95% confidence intervals were calculated, but only the rFEL network's interval is shown in the plot in the interest of avoiding clutter. The rFEL network performed statistically significantly better than the other algorithms, but the difference in performance between the other algorithms was not statistically significant. The MNIST sequence classification task was more challenging for the other algorithms than originally anticipated, with performance dropping rapidly after only a minor degree of non-stationarity was introduced. For example, using a non-stationary level of 0.06, 7,520 training samples from all sequences were presented, with the reduced classes being excluded from only the last 480 training samples. Under these conditions, the Simple Recursive Network's classification accuracy dropped from its baseline accuracy of 0.8962 to 0.5324, while the rFEL network's accuracy dropped from 0.9811 to 0.9347 over the same interval.

The motivation behind the FEL network is that the fixed layer neurons can form a meaningful sparse representation of the hidden layer signal. As such, it was expected that different FEL neurons would consistently react to different features within the input signal. To measure the FEL neuron affinity, the following data was collected. An 'affinity vector' was stored for each FEL neuron. For each test sample, the input signal was multiplied by the activation value of each FEL neuron and added to that neuron's affinity vector; if the FEL neuron was not activated, no change was made to its affinity vector. Once the evaluation process completed, the affinity vectors represented a weighted sum of input values to which the FEL neuron responded. By reconstructing a 32x32 pixel grayscale image from these weighted values (using the procedure described earlier), we can visually observe the input to which each FEL neuron responded. A collection of these affinity vectors is presented in Figure 11. It can be seen that the FEL neurons consistently reacted in response to specific patterns; one can observe varying levels of positive and negative responses to the digits 1 and 2, and in multiple cases the same neuron shows a positive response to one digit and a negative response to the other digit. This provides interesting insight into the actual representations captured by the sparse encoding of the FEL layer, further supporting the overall significant results observed in mitigating catastrophic interference.

V. Conclusions

In this paper, a novel approach for mitigating catastrophic forgetting in dynamic neural networks has been proposed. The approach is based on an enhancement to the Elman recurrent neural network, in which sparse coding of context (feedback) neurons is employed using a fixed expansion hidden layer. The revised RNN architecture exhibits significant improvements in retaining long-term memory in various

non-stationary settings, while incurring a modest increase in computational and storage requirements.

VI. Acknowledgment

This work was partially supported by the Intelligence Advanced Research Projects Activity (IARPA) via Army Research Office (ARO) agreement number W911NF-12-1-0017, and by NSF grant #CCF-1218492. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, the Department of the Army, the NSF, or the U.S. Government.

References

[1] B. Ans and S. Rousset. Avoiding catastrophic forgetting by coupling two reverberating neural networks. *Comptes Rendus de l'Académie des Sciences-Series III-Sciences de la Vie*, 320(12):989–997, 1997.

[2] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

[3] R. Coop and I. Arel. Mitigation of catastrophic interference in neural networks using a fixed expansion layer. In *Circuits and Systems (MWSCAS), 2012 IEEE 55th International Midwest Symposium on*, pages 726 –729, aug. 2012.

[4] Gregory Ditzler and Robi Polikar. Semi-supervised learning in nonstationary environments. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2741–2748. IEEE, 2011.

[5] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[6] J.L. Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7(2):195–225, 1991.

[7] R.M. French. Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks. *Connection Science*, 4(3/4):365–378, 1992.

[8] R.M. French. Dynamically constraining connectionist networks to produce distributed, orthogonal representations to reduce catastrophic interference. *network*, 1111:00001, 1994.

[9] R.M. French. Using pseudo-recurrent connectionist networks to solve the problem of sequential learning. In *Proceedings of the 19th Annual Cognitive Science Society Conference, NJ*, 1997.

[10] R.M. French and A. Ferrara. Modeling time perception in rats: Evidence for catastrophic interference in animal learning. In *Proceedings of the 21st Annual Conference of the Cognitive Science Conference*, page 173–178, 1999.

[11] Stephen Grossberg. Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural networks*, 1(1):17–61, 1988.

[12] M. Hattori. Dual-network memory model using a chaotic neural network. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, page 1–5, 2010.

[13] Yann Lecun and Corinna Cortes. The MNIST database of handwritten digits.

[14] H. Lee, A. Battle, R. Raina, and A.Y. Ng. Efficient sparse coding algorithms. *Advances in neural information processing systems*, 19:801, 2007.

[15] H. Lejeune, A. Ferrara, F. Simons, and J.H. Wearden. Adjusting to changes in the time of reinforcement: Peak interval transitions in rats. *Journal of Experimental Psychology: Animal Behavior Processes*, 23(2):211, 1997.

[16] O.M. Moe-Helgesen and H. Stranden. Catastophic forgetting in neural networks. 2005.

[17] R. Ratcliff. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological Review*, 97(2):285, 1990.

[18] A. Robins. Catastrophic forgetting in neural networks: the role of rehearsal mechanisms. In *Artificial Neural Networks and Expert Systems, 1993. Proceedings., First New Zealand International Two-Stream Conference on*, page 65–68, 1993.

[19] A. Robins and University of Otago. Artificial Intelligence Laboratory. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.