Mitigation of Catastrophic Interference in Neural Networks using a Fixed Expansion Layer

Robert Coop, Itamar Arel

***Abstract*—In this paper we present the fixed expansion layer (FEL) feedforward neural network designed for balancing plasticity and stability in the presence of non-stationary inputs. Catastrophic interference (or catastrophic forgetting) refers to the drastic loss of previously learned information when a neural network is trained on new or different information. The goal of the FEL network is to reduce the effect of catastrophic interference by augmenting a multi-layer perceptron with a layer of sparse neurons with binary activations. We compare the FEL network's performance to that of other algorithms designed to combat the effects of catastrophic interference and demonstrate that the FEL network is able to retain information for significantly longer periods of time with substantially lower computational requirements.**

Keywords: catastrophic forgetting, catastrophic interference, neural networks, non-stationary inputs, FEL, fixed expansion layer, non-stationary learning

I. Introduction

The problem of catastrophic interference (also referred to as catastrophic forgetting) in neural networks has been studied for over two decades by researchers from many disciplines such as machine learning, cognitive science, and psychology [8], [10]. In neural networks (and other connectionist architectures), catastrophic interference is the process by which a network "forgets" learned patterns upon being presented with new patterns. One can expect a neural network to have an inherent 'learning capacity' determined by the number of weights and neurons it contains. When this capacity is reached, learning new information will gradually interfere with a network's ability to recall existing information.

However, catastrophic interference is not caused by a network having reached its learning capacity. Instead, once the network has been trained on new patterns or is no longer being adequately prestented with inputs drawn from its entire observation space, drastic information loss occurs; the new information catastrophically interferes with the learned information even though there is plenty of learning capacity. Such scnearios are commonly encountered when non-stationary inputs are presented to the network.

Existing approaches for mitigating catastrophic forgetting include ...

In this paper we propose a novel approach for mitigating the fogetting phenomenon by augmenting existing MLP networks witha a dedicated sparse layer of binary neurons. By selectively activating neurons in this layer, latching of information is enabled with minor degradation exhibited as new inputs are provided to the network. The proposed approach is data-driven and computationally modest, facilitating large-scale implementations of such networks.

II. Conventional Mitigation of Catastrophic Forgetting

Many approaches to reducing the effect of catastrophic interference have been proposed, with varying levels of success. Notably, most exploration of the problem of catastrophic interference has been within the domain of autoassociative pattern learning, which has not addressed problems inherent with more general function approximation [9]. The most common schemes proposed in the literature can coarsely be grouped into the following categories.

A. Rehearsal methods

Rehearsal methods were among the first approaches to solving the problem of catastrophic interference; two such methods are the rehearsal buffer model [10] and sweep rehearsal [11]. Each method attempts to retain information about previously learned patterns by creating a buffer of some previously learned patterns; these buffered patterns are then periodically used for training during the learning of subsequent patterns. These early methods mitigated the effect of catastrophic interference somewhat, but required persistent storage of learned patterns.

B. Pseudorehearsal methods

Whereas rehearsal methods attempt to retain learned information by storing and rehearsing a set of examples, pseudorehearsal methods attempt to retain learned information without the requirement of pattern storage [12]. Pseudopatterns consisting of random input values are generated periodically during training. The pseudopattern is fed into the network and the network's output is recorded. After some number of subsequent training iterations, a previously generated pseudopattern is selected for pseudorehearsal. The pseudopattern is fed into the network, and the previously recorded output is used as a training target.

These pseudopatterns serve as approximate snapshots of the network's internal state at some time during the training process. As training proceeds, the network's internal state is being adjusted in order to recognize the currently viewed patterns. When pseudorehearsal is performed, the network's internal state is essentially being re-adjusted in order to be more like the snapshot of its prior internal state. This adjustment causes the network to be more likely to retain prior information, thus combating the catastrophic interference effects. However, the process of generating pseudopatterns and periodically retraining over these pseudopatterns increases the storage and computational requirements of the system. Moreover, analysis suggests that, in some networks, the effectiveness of pseudorehearsal is reduced when used with low-dimensional input or input patterns that are nearly (or completely) orthogonal [2].

C. Dual methods

Dual methods address catastrophic interference through attempting to separate that which is being learned from that which has already been learned; these methods are characterized by the explicit representation of short-term and long-term memory. Dual weight (e.g. [6], [7]) methods maintain two sets of weights for a single network architecture, while dual network (e.g. [4], [1],[5]) methods utilize entirely separate networks for the same purpose. While these type of methods have been shown to be somewhat effective, the computational and storage requirements are drastically increased; often these algorithms additionally perform rehearsal or pseudorehearsal (e.g. [6], [4], [5]).

D. Activation sharpening

Activation sharpening 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 this overlap [3]. The goal of activation sharpening is to gradually develop semi-distributed representations of patterns in the hidden layer of the network. This approach modifies this traditional feedforward process; the input pattern is fed forward, but then the activation of nodes in the hidden layer is 'sharpened' by increasing one or more of the hidden nodes with the largests activation values and decreasing the activation values of other hidden nodes. The difference between the original and sharpened activation values is immediately back-propagated to the input-hidden weights in order to train the network to produce a sharpened activation in the future. After this occurs, the input is fed forward and the error backpropagated as usual. This method does not significantly increase the memory requirements of the network, but it does result in a 50% increase in the computational requirements due to the additional half-backpropagation during each iteration.

III. The fixed expansion layer feedforward neural network

The motivation behind the fixed expansion layer (FEL) neural network is similar to the motivation for activation sharpening; the FEL network addresses the problem of representational overlap in a neural network by adding an additional 'expansion layer' into the network (pictured in figure 1). The weights for this layer are fixed at network initialization, with values such that the 'dense' signal contained in the hidden layer is expanded into a more 'sparse' signal contained in the FEL. During the feedforward process, the FEL neurons are 'triggered' in order to present a consistent sparse representation of the input pattern to the output layer. The sparseness of the FEL weights, combined with the triggered FEL neurons, protect the hidden layer weights from portions of the backpropagation error signal; this prevents the network weights from converging when learning new information and mitigates the effects of catastrophic interference.

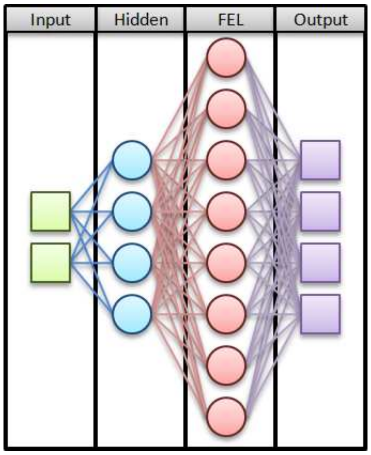


Figure 1. The FEL network

The fixed expansion layer can be used with many different feedforward neural network algorthms; the FEL only requires the additional steps of FEL weight initialization at network creation time and FEL neuron triggering during the feedforward process.

A. Weight initialization

The weights between the hidden layer and the FEL (pictured in figure 1) are set when the nerual network is created and are not updated during the learning process. These FEL weights must facilitate expansion of the signal contained in the activations of the hidden layer neruons into a more sparse representation of that signal in the FEL neurons. To acheive this, each FEL neuron is only connected to a portion of the neurons in the hidden layer. If each hidden neuron were fully connected to each FEL neuron (as in traditional feedforward neural network weighting), each FEL neuron's activation would be a function of the full hidden layer signal; the FEL layer's signal would be just as dense as that of the hidden layer. We therefore select only a portion of the hidden layer neurons to contribute to each FEL neuron.

To initialize the FEL weights, we first determine the number of hidden layer neurons that contribute to each FEL neuron (N_C) and the number of hidden layer neurons that will inhibit the activation of each FEL neuron (N_H). For each FEL neuron, we randomly select N_C contributory neurons and assign them a contributory weight (v_C) of 1. We then select N_H inhibitory neurons and assign them an inhibitory weight value (v_H) of $-\frac{v_C}{N_H}$. The selection of contributory and inhibitory neurons is performed such that each neuron in the hidden layer will contribute and inhibit the same number of FEL neurons.

B. Neuron triggering

During training, only a small portion of the FEL neurons (the 'triggered neurons') have nonzero activation values. Any hidden layer neuron connected to a triggered FEL neuron will receive a corrective training signal and consequently update all of its input layer weights; if all FEL neurons were triggered, then all hidden neurons would receive a training signal, and all input-hidden weights would be updated during each training iteration. We trigger some number (N_+) of neurons that have the largest activation value as well as some number (N_-) of neurons that have the smallest activation values. Intuitively, this can be thought of as selecting the neurons that strongly 'agree' or strongly 'disagree' with the hidden layer signal. These triggered neurons then have their activation value set to a constant value (v_+ or v_-), and all other FEL neurons have their activation value set to 0.

By setting the activation value of the triggered FEL neurons to specific values (as opposed to using their actual activation values), we are limiting the information that can be sent between the hidden layer neurons and the output layer neurons. In effect, we are dividing the learning process into two parts; the hidden layer weights are adjusted in order to create the sparse representation that will be most informative to the output layer, and the output layer weights are adjusted in order to interperet the sparse FEL signal into an accurate output.

IV. Experimental Results

A. Test setup

We performed a cluster classification test using a non-stationary training input. There are four clusters of two dimensional points, where each cluster has a mean and standard deviation and samples for that cluster are drawn from a Gaussian distribution. 50,000 training iterations are performed, followed by 1,000 testing iterations. The neural network is given the point coordinates as a two dimensional input, and produces a four dimensional output representing its classification. This problem is trivial when we train over samples drawn from each distribution for the entire training period; all algorithms tested are able to acheive 100% accuracy under this condition.

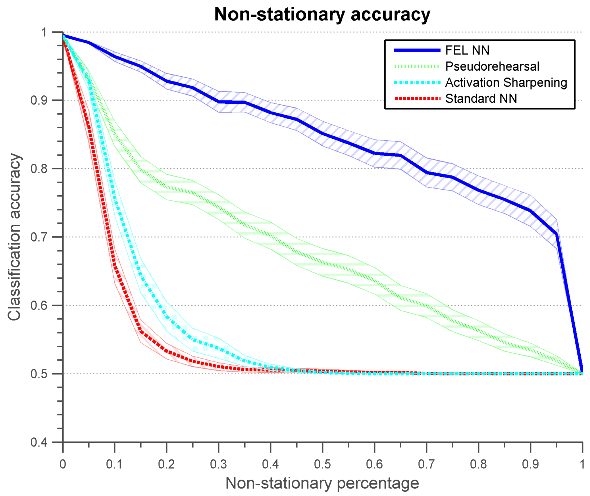
In order to test each algorithm's ability to resist catastrophic interference, we do not train over all four clusters during the training process. We present sample from all four clusters during the first portion of training, but then only present samples from two of the clusters (the 'primary' clusters) during the rest of training. During this second portion of training, no samples from the other two clusters (the 'restricted' clusters) are presented. Testing is still performed over all four clusters; the goal is to determine whether or not the network is able to retain information about all four clusters after being presented samples from only two clusters. The proportion of training that only uses two clusters (the 'non-stationary percentage') was adjusted in order to measure how well each algorithm performed under varying amounts of interference.

We tested the fixed layer feedforward neural network against a standard feedforward neural network, a network using activation sharpening, and a network using pseudorehearsal. Details of these network parameters are presented in table I.

For the FEL neural network, a FEL of 128 neurons was used. Each FEL received input from half of the hidden layer ($N_C = 4$, $N_H = 4$), with contributory weights of $v_C = 1$ and inhibitory weights of $v_H = -\frac{1}{4}$. For the neuron triggering, the $N_+ = 4$ neruons with the largest activation value and the $N_- = 1$ neuron with the smallest activation value were used. The positive trigger value was $v_+ = \frac{1}{2}$ and the negative trigger value was $v_- = -1$.

Table I
Neural network parameters

- All networks use 2 input neurons, 16 hidden layer neurons, and 4 output neurons.
- For activation sharpening, the 2 hidden layer neurons with the largest values were sharpened by a factor of $\alpha = 0.001$.
- Pseudorehearsal was performed by generating a new pseudopattern every 1000 training iterations. Every 100 training iterations, a random pseudopattern is selected and presented for training.



The shaded portion represents the 95% confidence interval for the accuracy. 'Non-stationary percentage' refers to the percent of the training that was performed using only two of the four possible clusters (i.e. a non-stationary percentage of 0 implies 50,000 training iterations using all four clusters, a non-stationary percentage of .75 implies that 12,500 iterations using all four clusters were performed followed by 37,500 iterations using only two of the clusters, etc.).

Figure 2. Classification accuracy

B. Results

For each value of the non-stationary percentage, 100 independent test runs were performed for each algorithm, and the results averaged in order to determine the mean accuracy, standard deviation, and the 95% confidence interval ($\alpha = 0.05$) for the mean accuracy. Networks were initialized with the same weights.

A plot of each network's accuracy is shown in figure 2, with some detailed values presented in table II. For all non-stationary percentages, the FEL shows the highest classifica-

%	Standard NN	Pseudorehearsal	Activation sharpening	FEL NN
0	1 (0)	1 (0)	1 (0)	1 (0)
0.25	0.52 (0.04)	0.76 (0.089)	0.55 (0.08)	0.92 (0.06)
0.5	0.51 (0.02)	0.66 (0.10)	0.50 (0.01)	0.85 (0.09)
0.75	0.5000 (0.00)	0.58 (0.08)	0.50 (0.00)	0.79 (0.10)
1	0.5 (0)	0.5 (0)	0.5 (0)	0.5 (0)

% is non-stationary percentage.
Main value is mean accuracy, std. deviation in parenthesis.

Table II
Classification accuracy detail

tion accuracy. Furthermore, the accuracy drops off at a roughly linear rate as the non-stationary percentage increases; the exponential decay in accuracy shown by the standard neural network is characteristic of catastrophic interference.

V. Conclusion and future work

conclusion:
Low resource usage, favorable accuracy
Future:
methods for: setting weights, node triggering
use in: ensemble methods

References

[1] B. Ans, S. Rousset, R.M. French, and S. Musca. Self-refreshing memory in artificial neural networks: learning temporal sequences without catastrophic forgetting. *Connection Science*, 16(2):71–99, 2004.

[2] M. Frean and A. Robins. Catastrophic forgetting in simple networks: an analysis of the pseudorehearsal solution. *Network: Computation in Neural Systems*, 10(3):227–236, 1999.

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

[4] 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.

[5] 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.

[6] G.E. Hinton and D.C. Plaut. Using fast weights to deblur old memories. In *Proceedings of the 9th Annual Conference of the Cognitive Science Society*, page 177–186, 1987.

[7] J.P. Levy and D. Bairaktaris. Connectionist dual-weight architectures. *Language and Cognitive Processes*, 10(3-4):265–283, 1995.

[8] M. McCloskey and N.J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The psychology of learning and motivation*, 24:109–165, 1989.

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

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

[11] 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.

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