Ensemble Learning in Fixed Expansion Layer Networks for Mitigating Catastrophic Forgetting

Robert Coop, *Student Member, IEEE*, Aaron Mishtal, *Student Member, IEEE*, and Itamar Arel, *Senior Member, IEEE*

***Abstract*— Catastrophic forgetting is a well-studied attribute of most parameterized supervised learning systems. A variation of this phenomenon, in the context of feedforward neural networks, arises when nonstationary inputs lead to loss of previously learned mappings. The majority of the schemes proposed in the literature for mitigating catastrophic forgetting were not data driven and did not scale well. We introduce the fixed expansion layer (FEL) feedforward neural network, which embeds a sparsely encoding hidden layer to help mitigate forgetting of prior learned representations. In addition, we investigate a novel framework for training ensembles of FEL networks, based on exploiting an information-theoretic measure of diversity between FEL learners, to further control undesired plasticity. The proposed methodology is demonstrated on a basic classification task, clearly emphasizing its advantages over existing techniques. The architecture proposed can be enhanced to address a range of computational intelligence tasks, such as regression problems and system control.**

***Index Terms*— Catastrophic forgetting, nonstationary inputs, sparse encoding neural networks.**

I. Introduction

The problem of catastrophic interference (also referred to as catastrophic forgetting [9]) in artificial neural networks has been studied over two decades by researchers from many disciplines, such as machine learning, cognitive science, and psychology [19], [24]. Many real-world applications, such as financial time series analysis and climate prediction, involve data streams that are either strictly nonstationary or can be considered piecewise stationary. It was shown that in mammals, 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) often at the expense of weakening existing associations [11], [16]. In parameterized supervised learning systems (like connectionist architectures), catastrophic interference is the process by which a network forgets learned patterns upon being presented with new patterns for a sufficiently long period. In such nonstationary settings, we 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 prior representations.

Catastrophic interference is, however, commonly not a result of a network reaching its learning capacity. Instead, once the network is trained on new patterns, or is no longer being adequately presented with inputs drawn from a prior observation distribution, drastic information loss occurs. The new information catastrophically interferes with learned representations even though there is a sufficient learning capacity. Such scenarios are commonly encountered when nonstationary input streams are presented to the network [5].

Many approaches with the goal of diminishing the impact of catastrophic interference were proposed in the literature, with varying levels of success [1], [2], [7], [12], [13], [17], and [25]. The vast majority of the schemes proposed do not pertain to online learning, but are rather based on batch-learning processes. In addition, most of the techniques require extensive memory resources, as they store prior configurations of the network as means of latching older representations. This paper proposes a novel approach for mitigating catastrophic forgetting by augmenting multilayer perceptron (MLP) networks with an additional sparsely encoded hidden layer specifically designed to retain prior learned mapping of inputs to outputs. Learning is completely incremental and minimal storage requirements are imposed.

The rest of this paper is structured as follows: Section II reviews existing solutions for mitigating catastrophic forgetting. In Section III, the fixed expansion layer (FEL) network is introduced. Section IV describes some improvements to the FEL approach using sparse coding. Section V proposes a method for improving the estimation problem in ensembles of FEL learners whereas in Section VI, simulation results for several catastrophic interference tasks are described. Finally, Section VII summarizes the conclusion.

II. Mitigating Catastrophic Forgetting in Nonstationary Settings

Mainstream exploration of the problem of catastrophic interference is within the domain of autoassociative pattern learning, which has not specifically addressed problems inherent with more general function approximation [20]. The following outlines the most commonly used schemes proposed in the literature.

A. Rehearsal Methods

Rehearsal methods are among the first approaches aimed at addressing the problem of catastrophic interference. Two such

Manuscript received May 16, 2012; revised April 25, 2013; accepted May 16, 2013. This work was supported in part by the Intelligence Advanced Research Projects Activity via Army Research Office under Grant W911NF-12-1-0017, and by NSF under Grant CCF-1218492.

The authors are with the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN 37916 USA (e-mail: coop@eecs.utk.edu; amishtal@eecs.utk.edu; itamar@eecs.utk.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2013.2264952

methods are as follows: 1) the rehearsal buffer model [24] and 2) sweep rehearsal [25]. Each method attempts to retain information about formerly learned patterns by maintaining a buffer of previously observed inputs. These buffered patterns are then periodically used for training during the learning of subsequent patterns. Such early methods mitigated the effect of catastrophic interference somewhat, but required persistent storage of learned patterns and introduced challenges with respect to the correct balance between new patterns and buffered ones. Therefore, substantial fine tuning of parameters, such as buffer sizes and replay frequency, are required.

1) Rehearsal Buffer Model: The rehearsal buffer model operates as follows: assuming there are M patterns to be learned, we create a rehearsal buffer containing a small subset of m patterns (e.g., initially the first pattern through the mth pattern, with typical values of m being around 4). Next, the neural network is trained over each pattern in the buffer. Then, the network is presented with each pattern in the buffer sequentially, and the entire buffer is looped over N times. The constant N is specified by the experimenter; there is no explicit relationship between M and N. Upon completing N loops through the buffer, the first item in the buffer is replaced by the subsequent pattern to be learned. Therefore, the buffer would first contain items in the range $[1, m]$, the training process loops over these items N times, then the buffer is updated to include items in the range $[2, m + 1]$. This process continues until all M patterns are learned.

2) Sweep Rehearsal: Sweep rehearsal is similar to the rehearsal buffer model, however it uses a dynamic training buffer rather than a fixed one. Given M patterns and a buffer size of m patters, we must first learn at least $(m - 1)$ patterns before the dynamic buffer comes into play. Once a sufficient number of patterns are presented to the network (with a pattern considered to be learned when trained until the estimation error is below a given threshold), the process of learning a new pattern proceeds as follows. Let the new pattern to be learned be denoted as x_i. The training buffer is created by combining pattern x_i with $(m - 1)$ previously learned patterns (selected at random), and the network is trained by presenting each pattern in the buffer once. Where the rehearsal buffer model would sweep through this same buffer N times, sweep rehearsal, however, constructs a new buffer containing x_i and $(m - 1)$ randomly selected from previously learned patterns. This is performed during each epoch, where an epoch is one presentation of each pattern in the buffer. A new buffer is created and used for training until pattern x_i is considered learned, at which point the process is repeated for pattern x_{i+1}, and so on. In practice, sweep rehearsal delivers improved performance when compared with the rehearsal buffer model [25], however it still requires substantial tuning and is inherently limited in its capacity to retain long-term memory.

B. Pseudorehearsal Methods

Whereas rehearsal methods attempt to retain learned information by storing and rehearsing previously learned patterns, pseudorehearsal methods attempt to latch onto learned information without the requirement of pattern storage [26]. Instead of using previously learned patterns for rehearsal, 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.

Pseudorehearsal can use the rehearsal buffer model or sweep rehearsal as the base learning mechanism. Random pseudopatterns are, however, used instead of actual patterns that were previously learned. To use pseudorehearsal with the sweep rehearsal approach, we would construct our sweep buffer by selecting the nth pattern to be learned and generating $(m - 1)$ pseudopatterns [as opposed to randomly selecting $(m - 1)$ previously learned patterns] before each epoch. During each epoch, the network is trained over the sweep buffer once. Then, we construct another sweep buffer containing the nth pattern and newly generated pseudopatterns. This process repeats until the nth pattern is learned sufficiently, then, repeat using the $(n + 1)$th pattern, and so on.

Consider, the problem of learning binary patterns with an autoassociative neural network. The training data is given as input–output pairs (x_i, y_i), where x_i is the ith input and y_i is the desired ith output. The function to be approximated is $f(x_i) = y_i$. For the autoassociative problem, $x_i = y_i$ and $f(\cdot)$ are the identity function. The function approximation performs by the autoassociative neural network $h(x_i) = \hat{y}_i$, where $\hat{y}_i$ is the output of the neural network (and we desire $h(x_i) = \hat{y}_i \approx y_i$).

This scheme involves generating a pseudopattern p_i by randomly setting each bit to either zero or one with equal probability. We compute the output of the network over this pseudopattern and store the result as q_i (i.e., we compute $h(p_i) = q_i$). For pseudopatterns, instead of training to achieve the goal $q_i \approx p_i$, we use the value of q_i as the actual training target. Therefore, our sweep buffer would contain our pattern to be learned, (x_i, y_i) as well as a number of pseudopatterns (p_i, q_i).

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 to recognize the currently viewed patterns. When pseudorehearsal is performed, the network's internal state is essentially being readjusted 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. The process of generating pseudopatterns and periodically retraining over these pseudopatterns, however, increases the storage and computational requirements of the system. In addition, analysis suggests that, in some networks, the effectiveness of pseudorehearsal degrades when used with low-dimensional input or input patterns that are nearly (or completely) orthogonal [7].

C. Dual Methods

Dual methods address catastrophic interference by means of attempting to separate that which is being learned from that which was already learned; these methods are characterized by the explicit representation of short-term and long-term memory. Dual-weight methods [13], [17] maintain two sets of weights for a single-network architecture, whereas dual-network methods [2], [10], [12] use entirely separate neural networks. Both approaches use one resource (a set of weights or a network) for storing long term, slowly changing information, and use the other resource for storing short-term, quickly changing information. While these type of methods are shown to be somewhat effective, the computational and storage requirements are drastically increased; often these algorithms additionally perform rehearsal or pseudorehearsal (e.g., [10], [12], [13]).

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 such overlap [8]. The goal of activation sharpening is to gradually develop semidistributed 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 nodes in the hidden layer is sharpened by increasing one or more of the hidden nodes with the largest activation values and decreasing the activation values of all the other hidden nodes. The difference between the original and sharpened activation values is immediately backpropagated to the input-hidden weights (as if it is an error signal) 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 because of the additional half-backpropagation procedure during each iteration. The additional backpropagation procedure becomes increasingly expensive to perform as the neural network grows in size, which has scalability implications.

III. FEL Neural Network

The motivation behind the FEL neural network is similar to that of activation sharpening, with the exception that FEL inherently supports an incremental, online learning process, and exploits sparse encoding to latch onto previously learned input/output mappings. The FEL network addresses the problem of representational overlap in a feedforward neural network by exploiting an augmented MLP architecture that includes the addition of an expansion hidden layer to the network, as shown in Fig. 1. The weights for this layer are fixed during network initialization, with weight values chosen such that the dense signal contained in the hidden layer is expanded into a more sparse signal represented by the FEL. In particular, some of the fixed weights between the hidden layer and FEL are excitatory while others are inhibitory. Sparsity, thus serves as means of mitigating forgetting of older information by selectively gating weight update signal propagation through the network.

During the feedforward phase, the FEL neurons are triggered to present a consistent sparse representation of the input pattern to the output layer (see Fig. 2). The sparsity of the FEL weights, combined with the triggered FEL neurons, protect the input-to-hidden layer weights from portions of the backpropagation error signal (see Fig. 3), thus preventing the network weights from changing drastically when exposed to new information that mitigates the effects of catastrophic interference.

In this paper, while MLP networks are treated, the FEL can be used by many other neural network architectures and training algorithms. Employing the FEL requires the additional step of fixed weight initialization at the time of network creation and defining the neural triggering conditions, as described in the following.

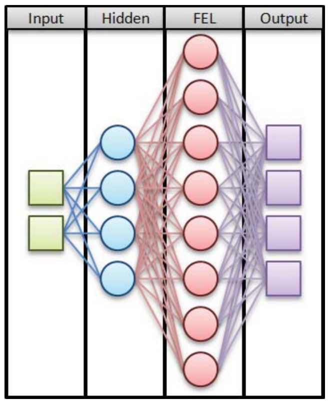

Fig. 1. FEL neural network architecture. A sparse-encoded hidden layer resides between the hidden and the output layers.

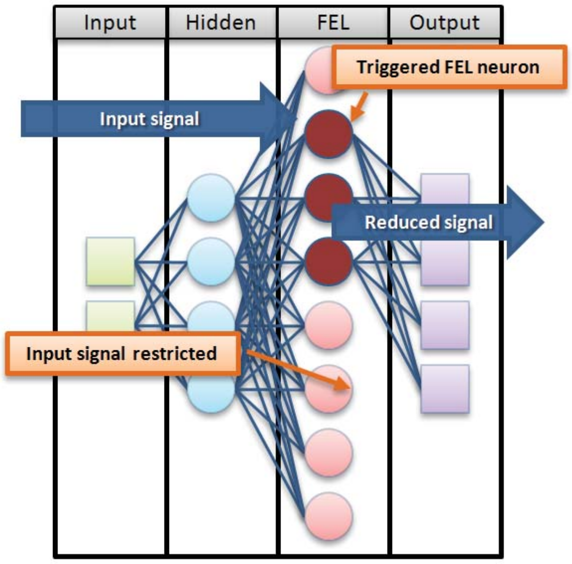

Fig. 2. Feedforward signal flow in the FEL neural network.

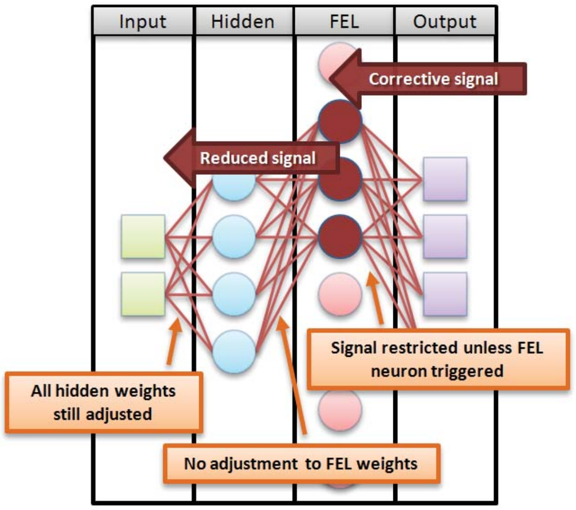
Input
Hidden
FEL
Output
Corrective signal
Reduced signal
Signal restricted unless FEL neuron triggered
All hidden weights still adjusted
No adjustment to FEL weights

Fig. 3. Error back-propagation in the FEL neural network.

A. Weight Initialization

The weights between the hidden layer and the FEL (Fig. 1) are set when the neural network is created and remain unchanged during the learning process. These FEL weights must facilitate expansion of the signal contained in the activations of the hidden layer neurons into a sparse representation by the FEL neurons. To achieve this, each FEL neuron is only connected to a subset of neurons in the hidden layer. If each hidden neuron are 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. Therefore, the FEL layer's signal would be just as dense as that of the hidden layer. Therefore, we select only a subset of the hidden layer neurons to contribute to each FEL neuron.

To initialize the FEL weights, we specify the number of hidden layer neurons that are excitatory with respect to each FEL neuron's activation (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 hidden neurons and assign these neurons a positive weight value (υ_C) of 1. Next, N_H inhibitory neurons are selected and assigned negative weight value (υ_H) of $-\upsilon_C/N_H$. Selection of excitatory and inhibitory neurons is performed such that each neuron in the hidden layer will excite and inhibit the same number of FEL neurons, thus ensuring balanced mapping of signals between the two hidden layers of the network.

B. Neuron Triggering

During training, only a small subset of the FEL neurons (i.e., the triggered neurons set) have nonzero activation values. Any hidden layer neuron connected to a triggered FEL neuron will receive a corrective training signal through the back-propagation process and therefore update all of its input layer weights. If all FEL neurons are to be triggered, then all hidden neurons would receive a corrective training signal, and therefore all input-to-hidden layer weights would be updated during each training epoch. In contrast, in the proposed FEL network only some number (N_A) of neurons are triggered—specifically those that have the highest activation values. In addition, a number (N_D) of neurons that have the lowest activation values are activated. Intuitively, this can be thought of as selecting the neurons that strongly agree or disagree with the hidden layer signal. The degree of agreement or disagreement is determined by the excitatory and inhibitory weights between the hidden layer neurons and the FEL neurons. These excitatory and inhibitory triggered neurons have their activation values set to a positive (υ_p) or negative (υ_n) constant, respectively, and all other FEL neurons have their activation values set to zero.

Through setting the activation levels of the triggered FEL neurons to specific values (as opposed to using their actual activation values), we are effectively limiting the information that can be propagated between the hidden layer neurons and output layer neurons. In effect, we are partitioning the learning process into two parts as follows: 1) the hidden layer weights are adjusted to create the sparse representation that will be most informative to the output layer; and 2) the output layer weights are adjusted to map the sparse FEL signal into an accurate representation at output layer.

C. Ensembles of FEL Networks

When examining the results from experiments with a single-FEL network, it is observed that when FEL networks exhibit some degree of forgetting, they tend to do so at different regions of the input space. This observation suggests that a sufficiently diverse set of FEL networks, trained as an ensemble of learners, may effectively overcome the limitation of any single-FEL network. Ensemble learning has been extensively studied in recent years, with many well-understood algorithms, such as Boosting [22] and Bagging [3], being proposed, as well as methods designed for nonstationary environments [21]. The key idea behind ensemble learning is to establish a set of trained learners such that their expected aggregate decision is more accurate than that of any individual learner. For this desired property to hold, the learners need to be diverse.

A particular family of ensemble training algorithms, which can be easily exploited in the context of neural network learners, is negatively correlated learning (NCL) [27], [28]. These algorithms involve modifying the cost function of each learner so as to promote diversity between the ensemble members. In particular, the mean squared error (MSE) cost function is augmented by an additional term that reflects a negative correlation between each learner's error (i.e., difference between target and output) and the average error of the ensemble. This inherently forces differentiation between the learners thus, yielding an improved overall level of accuracy. While NCL is a solid means of guaranteeing diversity, it is not optimized for classification problems. Section V outlines an alternative, information-theoretic cost term, that offers richer diversity between the learners.

D. Computational Complexity and Limitations

The computational complexity of the FEL algorithm does present some additional computational and memory requirements. During the feed-forward step, an additional computation must be performed for the neuron triggering; this can be completed by sorting the FEL neurons by activation value. The FEL layer itself requires some additional memory for storage. This additional storage requirement is, however, negligible when compared with the rehearsal and pseudorehearsal requirements for storing arbitrary numbers of patterns (or pseudopatterns).

IV. Improving the FEL Algorithm Using a Sparse Coding Approach

Sparse coding can be used as an improved method for determining the FEL activation neurons and their values. In such framing, the activations of the hidden layer denote the dense signal, the activations of the expansion layer represent the sparse signal, and the FEL weights act as the basis for the transformation. We seek to minimize the L_1 norm of the FEL neurons while at the same time retaining key information from the hidden layer neurons. This is equivalent to the following optimization formulation:

$$\text{minimize}_x f(x) \equiv ||y - Ax^2|| + \gamma\, ||x||_1 \tag{1}$$

where y is 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 that acts to balance the desire for a sparse FEL activation and retention of information from the hidden layer.

This problem can be efficiently solved using the feature-sign search algorithm introduced in [15]. Equation (1) can equivalently be written as follows:

$$\text{minimize}_x f(x) \equiv ||y - Ax^2|| + \gamma \sum_{i=1}^{N} |x_i| \tag{2}$$

where N is the number of elements in x. 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 (QP), which can be solved analytically and efficiently. The feature-sign search algorithm tries to search for the signs of the coefficients x_i; given a guess about the signs of these coefficients, the resulting unconstrained QP can be efficiently solved. The algorithm systematically refines the guess if it turns out to be initially incorrect.

This algorithm operates by maintaining an active set of potentially nonzero coefficients in x and the sign (positive or negative) of these values; all other members of x are assumed to be zero. Given a guess about the current active set and corresponding signs, an analytical solution to the minimization can be computed. With the latter, we can improve the guess and repeat the process until a termination condition is met. Each such step reduces the objective function $f(x)$, and the process is guaranteed to converge to the optimal solution. The algorithm is shown in Table I.

The framing of the FEL network within the sparse coding domain and the application of the feature-sign search algorithm lead to significant advantages. Through applying this approach, we are able to eliminate eight constants ($N_C, N_H, \upsilon_c, \upsilon_h, N_A, N_D, \upsilon_p, \upsilon_n$), and replace these with a single constant (γ). Beyond offering a more firm theoretical grounding to the FEL approach, this improvement results in significantly higher accuracy over the original approach, as discussed in Section VI.

V. Ensembles of FEL Networks

A. Jensen–Shannon Divergence

The Jensen–Shannon divergence (JSD) [18] provides an information-theoretic measure of the similarity between two probability distributions. Given two discrete distributions P and Q, their JSD is given by

$$\text{JSD}(P \parallel Q) = \frac{1}{2} D_{\text{KL}}(P \parallel M) + \frac{1}{2} D_{\text{KL}}(Q \parallel M) \tag{3}$$

where $D_{\text{KL}}(P \| Q)$ is the Kullback-Leibler divergence between P and Q

$$D_{\text{KL}}(P \| Q) = \sum_i P(i) \ln \frac{P(i)}{Q(i)} \tag{4}$$

and M is the mixture distribution defined by

$$M = \frac{1}{2}(P + Q). \tag{5}$$

Further, the quantity $\sqrt{JSD(P \parallel Q)}$ is a metric [6] that can serve as a diversity measure between two ensemble members.

When a neural network is used for classification and has several output nodes, each node can be viewed as representing a level of confidence that the given input belongs to that class. We can force the network to produce a discrete probability distribution over the classes that corresponds to these confidence levels by normalizing its outputs via the softmax function

$$\bar{h}_{jk}(x_i) = \frac{\exp(h_{jk}(x_i))}{\sum_{l=1}^{N} \exp(h_{jl}(x_i))} \tag{6}$$

where $h_{jk}(x_i)$ is the kth output of ensemble member j in response to input x_i, and the summation is performed over each dimension of the ensemble member's output (i.e., N is the number of outputs for the jth ensemble member). This will map the outputs to points in a probability space such that their sum is equal to one. It will also cause relatively large output values to retain more of the probability mass.

Within the context of a neural network ensemble with network outputs normalized as outlined above, we can use the JSD metric as a measure of diversity between individual network hypotheses. In addition, by maximizing the JSD value between a given network's output and the ensemble average, we can increase the diversity within an ensemble. To accomplish this, we replace the traditional network error function with a convex sum between the network error and the negative square root of the JSD between its normalized

TABLE I
FEATURE-SIGN SEARCH ALGORITHM [15]

```
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 $\theta$ 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 := $ 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.
```

output and the normalized ensemble average, which yields the following new error (cost) function

$$e_{jk}(x_i) = (1-\gamma)(y_{ik} - h_{jk}(x_i))^2 - \gamma\sqrt{\text{JSD}(\bar{h}_j(x_i) \parallel \bar{\mathcal{H}}(x_i))} \quad (7)$$

where $\bar{\mathcal{H}}(x_i)$ is the softmax-normalized output of the entire ensemble, y_{ik} is the kth target output, and $0 < \gamma < 1$ is a constant.

The derivative of this additional term with respect to a particular (prenormalized) network output is

$$\frac{\partial e_{jk}}{\partial h_{jk}} = -\left\{(1-\gamma)(y_{ik} - h_{jk}(x_i)) + \alpha_{jk}\left[(1 - \bar{h}_{jk}(x_i)\beta_{jk} - \sum_{l\neq k}\bar{h}_{jl}(x_i)\beta_{jl}\right]\right\} \quad (8)$$

where

$$\alpha_{jk} = \frac{\bar{h}_{jk}(x_i)}{\sqrt{2\text{JSD}(\bar{h}_j(x_i) \parallel \bar{\mathcal{H}}(x_i))}} \quad (9)$$

and the terms β_{jk} and β_{jl} are given by

$$\beta_{jn} = \ln\left(\frac{2\bar{h}_{jn}(x_i)}{\bar{h}_{jn}(n) + \bar{\mathcal{H}}_n(x_i)}\right) + \frac{1}{N}\ln\left(\frac{2\bar{\mathcal{H}}_n(x_i)}{\bar{h}_{jn}(n) + \bar{\mathcal{H}}_n(x_i)}\right) \quad (10)$$

where n in β_{jn} is replaced by either k or l, depending on whether we are referring to β_{jk} or β_{jl} (as specified in 8). The choice of γ determines the balance between the two goals of the cost function, such as minimizing the MSE (i.e., approximating a maximum likelihood estimator) while also promoting diversity between the learners.

B. Estimating Classification Confidence

The output of each learner in an ensemble can be viewed as a point estimator, with no explicit information about the uncertainty of the estimation provided. If each learner are to produce an estimate of its deviation from the target (i.e., a reflection of its confidence interval) for each output, it would be possible to weight the learners inversely proportional to their estimated error such that the more confident networks will weigh more than the less confident ones. The use of FEL networks in estimating error intervals is further motivated by their inherent ability to retain long-term memory, thus improving their error estimation. In approximating an N-dimensional function, we also require each neural network to estimate its own deviation from the target for each output, such that the output of a learner is as follows:

$$h_j(x_i) = \begin{bmatrix} \hat{y}_{i1}^{(j)} \\ \hat{e}_{j1}(x_i) \\ \hat{y}_{i2}^{(j)} \\ \hat{e}_{j2}(x_i) \\ \vdots \\ \hat{y}_{iN}^{(j)} \\ \hat{e}_{jN}(x_i) \end{bmatrix} \quad (11)$$

where h_j is trained with the goals that $\hat{y}_i^{(j)} \approx \hat{y}_i$ and the absolute deviation is used as the confidence target, such that $\hat{e}_{jk}(x_i) \approx \left|y_{ik} - \hat{y}_{ik}^{(j)}\right|$. A similar framework is studied in the context of an ensemble of Bayesian learners [23].

Using this methodology, we can set w_{jk}, the weight assigned to the kth output of the jth ensemble member proportional to its estimated absolute error. As such, we have $w_{jk} \propto \hat{e}_{jk}(x_i)^{-1}$, which will lower the overall error in $\mathcal{H}$ as long as our estimation of $e_{jk}(x_i)$ is reasonably accurate. Hence, the key to this weighting scheme surpassing the accuracy of simply averaging the learner's outputs is that a sufficiently high correlation coefficient be observed between the estimated absolute errors and the actual ones.

Calculating w_{jk} from $\hat{e}_{jk}(x_i)$ is done by taking a Boltzmann weighted mean over all the error estimates for each ensemble member, calculated as follows

$$w_{jk} = \frac{\exp\left(-\frac{\hat{e}_{jk}(x_i)}{\tau}\right)}{\sum_{l=1}^{N}\exp\left(-\frac{\hat{e}_{jl}(x_i)}{\tau}\right)}. \tag{12}$$

VI. Simulation Results and Analysis

We perform simulations to compare the ability of the FEL network to resist catastrophic forgetting in comparison with several other algorithms. We study the FEL network in isolation as well as in the context of ensemble learning on several different tasks.

A. Traditional Catastrophic Forgetting Task (Autoassociative Binary Pattern Reconstruction)

1) Task Description: For this task, we compare FEL network performance to two of the commonly used approaches to mitigation of catastrophic forgetting. We also show the performance of a standard MLP feedforward neural network for reference. Mainstream exploration of the problem of catastrophic interference is within the domain of autoassociative pattern learning [20]; this task compares the ability of a network to retain previously learned information after learning new information.

The autoassociative binary pattern reconstruction test is performed as follows. Initially, the network is trained over a set of 20 patterns, where each pattern consists of 32 binary values. The goal of the network is to recreate the input provided; the network has 32 real-valued outputs that do not get rounded. The network is trained over all 20 patterns until the MSE for the set is less than 0.06. Once the network has learned the 20 base items, we present a new pattern to the network. The network is trained once using this intervening item, and we then measure the MSE over the original set of base items to determine how much of the original information is retained.

2) Networks Tested: We evaluate the FEL neural network against a standard MLP feedforward neural network, a network using activation sharpening, and a network using pseudorehearsal. All networks have 32 input neurons, 16 hidden layer neurons, and 32 output neurons. For activation sharpening, the two hidden layer neurons with the largest values are sharpened by a factor of $\alpha = 0.001$. Pseudorehearsal is performed by generating 32 pseudopatterns after original training over the binary patterns. Every time an interleaving pattern is learned, a random pseudopattern is selected and presented for training.

In the FEL neural network, 128 neurons are used in the sparse (fixed) layer. Each FEL received inputs from half of the hidden layer nodes (i.e., $N_C = 4$ and $N_H = 4$), with

TABLE II
Pattern Reconstruction Accuracy for the Traditional Catastrophic Forgetting Task

Intervening Items	Standard MLP	Pseudo-rehearsal	Activation sharpening	**FEL NN**
1	0.930	0.934	0.926	**0.941**
3	0.922	0.931	0.917	**0.936**
5	0.918	0.930	0.914	**0.932**
7	0.914	0.927	0.909	**0.929**
10	0.914	0.926	0.911	**0.926**

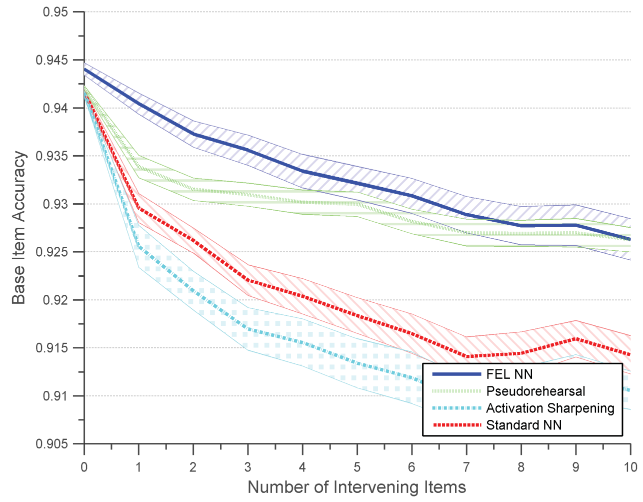

Fig. 4. Pattern reconstruction accuracy comparison for the traditional catastrophic forgetting task. Each line represents the mean classification accuracy, with the shaded proportion representing the 95% confidence interval.

excitatory weights of $v_C = 1$ and inhibitory weights of $v_H = -0.25$. For the neuron triggering, the $N_C = 4$ neurons with the largest activation value and the $N_H = 1$ neuron with the smallest activation value are used. The positive trigger value is $v_p = 0.5$ and the negative trigger value is $v_n = -1$.

3) Simulation Results: Each network is used in 100 independent test runs, and the results used to determine the mean accuracy, standard deviation, and the 95% confidence interval for the mean accuracy.

A plot of these values is shown in Fig. 4, with some detailed results shown in Table II. The FEL network performs significantly better than the standard MLP network and the network using activation sharpening and is slightly better than the network using pseudorehearsal. These results show that the FEL network performs well in the domain typically used for testing catastrophic forgetting.

B. Single-Learner Nonstationary Gaussian Distribution Classification Task

2-D inputs are drawn from four Gaussian distributions, where each has a different mean. The process involves performing a total of 50,000 training iterations, followed by 1000 testing iterations. A neural network is given the two-dimensional input (x and y values) as input, and produces a 4-D output representing the posterior distribution over the classes. This problem is considered trivial when the sample

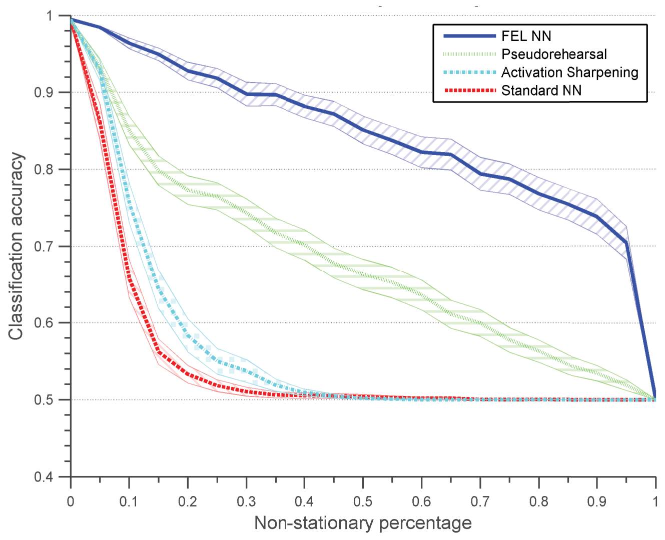

Fig. 5. Classification accuracy comparison for the single-learner nonstationary Gaussian distribution classification task. Each line represents the mean classification accuracy, with the shaded proportion representing the 95% confidence interval. Nonstationary percentage: the percent of total training iterations that are performed using only samples drawn from the primary clusters.

distribution is fixed and does not change during the entire training period. All algorithms evaluated are able to achieve 100% accuracy under such conditions.

When nonstationary conditions are, however, assumed, the performance of most algorithms degrade. To study each algorithm's ability to mitigate catastrophic interference, the sample distribution varies over the training period such that samples are presented from all four Gaussian clusters during the initial phase of the training process, followed by a duration of time in which samples are drawn only from two of the Gaussian clusters (i.e., the primary clusters). During this latter phase of the training, no samples drawn from the other two Gaussian clusters (i.e., the restricted clusters) are presented. Testing is still performed over both the primary and the restricted clusters with the goal being to determine whether or not the network is capable of retaining information about the restricted clusters upon being presented with multiple samples drawn only from the primary clusters. The proportion of training iterations that pertain to the primary clusters (i.e., the nonstationary percentage) is adjusted to measure how well each algorithm performed under varying amounts of interference.

Networks are tested using the same parameters as in the previous section (using two input and four output nodes).

For each value of the nonstationary percentage, 100 independent test runs are performed for each algorithm, and the results used to determine the mean accuracy, standard deviation, and the 95% confidence interval for the mean accuracy.

A plot of each network's accuracy is shown in Fig. 5, with some detailed values shown in Table 5. For all nonstationary percentages, the FEL shows the highest classification accuracy. Furthermore, the accuracy drops off at a roughly linear rate as the nonstationary percentage increases in contrast to the exponential decay in accuracy demonstrated by standard MLP, which is characteristic of catastrophic interference. Fig. 6 provides more detail for a single-test run (with a nonstationary percentage of 75%). The standard MLP loses all ability to classify samples from the restricted clusters, whereas the FEL network only misclassifies samples that lie near the edges of the restricted clusters.

TABLE III
CLASSIFICATION ACCURACY FOR VARIOUS CATASTROPHIC INTERFERENCE MITIGATION SCHEMES COMPARED WITH THE FEL NEURAL NETWORK

Non-stationary percentage	Standard MLP	Pseudo-rehearsal	Activation sharpening	**FEL NN**
0.00	1.00	1.00	1.00	**1.00**
0.25	0.52	0.76	0.55	**0.92**
0.50	0.51	0.66	0.50	**0.85**
0.75	0.50	0.58	0.50	**0.79**
1.00	0.50	0.50	0.50	**0.50**

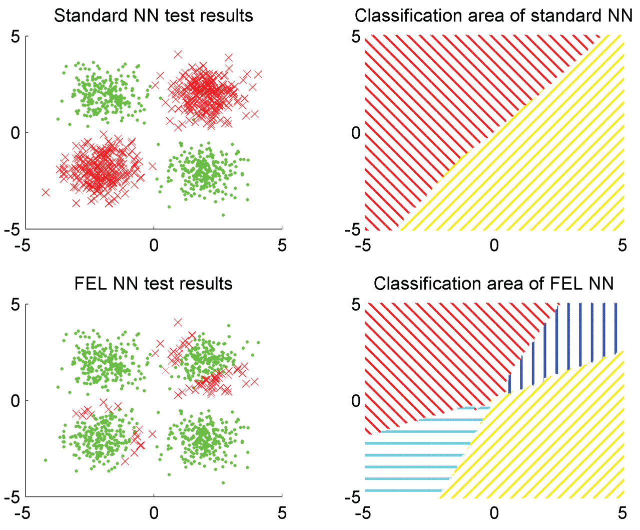

Fig. 6. Classification region comparison between a FEL neural network (bottom) and a standard MLP neural network (top). The primary clusters are centered at (-2, 2) and (2, -2). The standard MLP loses the classification region for both of the restricted clusters, whereas the FEL network only loses the borders of the restricted clusters. Left side figures: results for the test set, with green circles: correct classifications and red X's: incorrect classifications. Right side figures: entire input space and the classification that each network would make for a point in each region.

C. Ensemble Learning Nonstationary Gaussian Distribution Task

To compare ensemble techniques, we evaluate various schemes for composing multiple FEL neural networks into an ensemble. Each ensemble is composed of seven FEL neural networks, using the same parameters as in the single-learner case. We compare JSD as a diversification term with and without weighting the learners proportionally to their estimated error. Furthermore, both a basic ensemble that uses the mean value of member outputs and an ensemble trained using NCL are considered. The negative correlation penalty term used is $\gamma = 0.5$ (as in [4]), and the JSD term is weighted using a convex with the MSE term, where $\gamma = 0.8$. The temperature parameter in the estimated error weighting is $\tau = 0.1$.

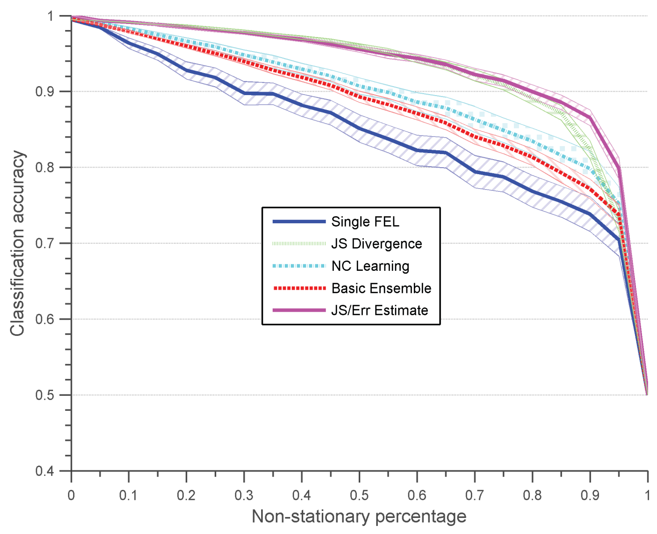

Fig. 7. Classification accuracy in ensembles of FEL networks applied to a basic clustering task. Each line represents the mean classification accuracy, with the shaded proportion representing the 95% confidence interval. Nonstationary percentage refers to the percent of total training iterations that are performed using only samples drawn from the primary clusters.

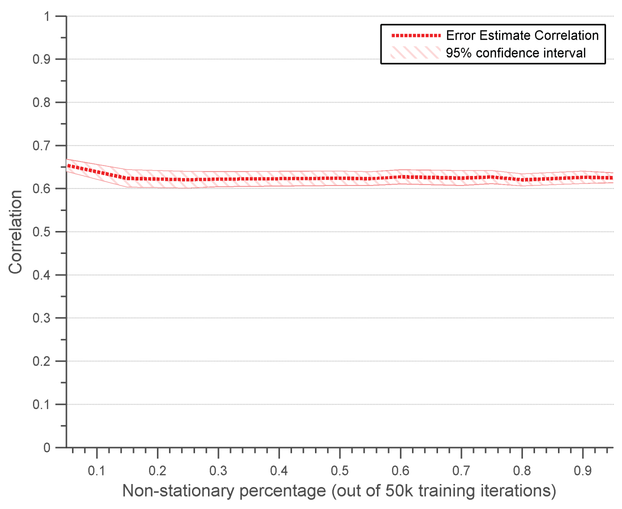

Fig. 8. Correlation coefficient between the estimated error and the actual error (used for error weighting).

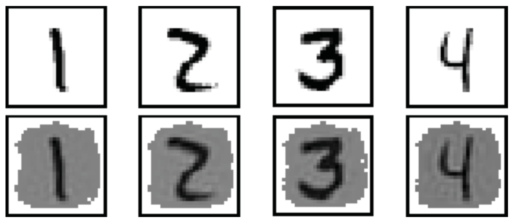

Fig. 9. Comparison of the original 1024-D handwritten digits (top) and the PCA reduced and reconstructed 128-D digits (bottom).

TABLE IV
CLASSIFICATION ACCURACY FOR VARIOUS ENSEMBLE TECHNIQUES

Non-stationary percentage	Basic Ensemble	Negative Correlation Learning	Jensen-Shannon Divergence Diversification	**JSD with Estimated Error Weighting**
0.00	1.00	1.00	1.00	**1.00**
0.25	0.95	0.96	0.98	**0.98**
0.50	0.89	0.91	0.96	**0.96**
0.75	0.83	0.85	0.91	**0.91**
0.90	0.77	0.80	0.82	**0.87**
1.00	0.50	0.50	0.50	**0.50**

For each value of the nonstationary percentage, 100 independent test runs are performed for each algorithm, and the results used to determine the mean accuracy, standard deviation, and the 95% confidence interval for the mean accuracy.

A plot of the accuracy obtained using an ensemble of FEL neural networks is shown in Fig. 7, with some detailed values shown in Table IV. Results from the ensemble technique comparison show that significant accuracy gains can be made using the FEL neural network in an ensemble setting. All ensemble techniques perform significantly better than a single learner, with the most significant accuracy differences occurring at higher levels of nonstationarity. To examine the possible performance of error estimated weighting, we investigate the correlation between the estimated error and the actual error. The results are shown in Fig. 8. The correlation level remains significantly high across the entire range of nonstationarity values and, correspondingly, the JSD ensemble using estimated error weighting performs better than other approaches at higher levels of nonstationarity.

D. Nonstationary MNIST Classification Task With Four Digits

The previous tasks illustrate the viability of the FEL network when considering the traditional domain over which catastrophic interference is measured and when considering a simple Gaussian classification problem. This task shows that the FEL network is applicable to more complex tasks and that the FEL network's accuracy is significantly improved by the addition of the feature-sign search algorithm.

This task involves the classification of digits taken from the MNIST database of handwritten digits [14]. Digits 1, 2, 3, and 4 are used for this task. The original MNIST digits are 32×32 pixel grayscale images, with each image consisting of 1024 pixels taking on integer values from 0 (white) to 255 (black). We preprocess the MNIST data by shifting the pixel values to be a real number between zero and one, centering the data, and using principal component analysis (PCA) to reduce the 1024-dimensional data to 128 dimensions. This dimensionality reduction captures approximately 94% of the variance of the original data. Through taking the Moore–Penrose pseudoinverse of the principal component matrix, we can reconstruct the reduced data to obtain a visual representation of the amount of information present in the reduced dataset. This representation is shown in Fig. 9.

To use the MNIST data to study the mitigation of catastrophic interference, we use a similar approach to that used in the Gaussian classification task. The training period is divided into two phases as follows: 1) first phase—we

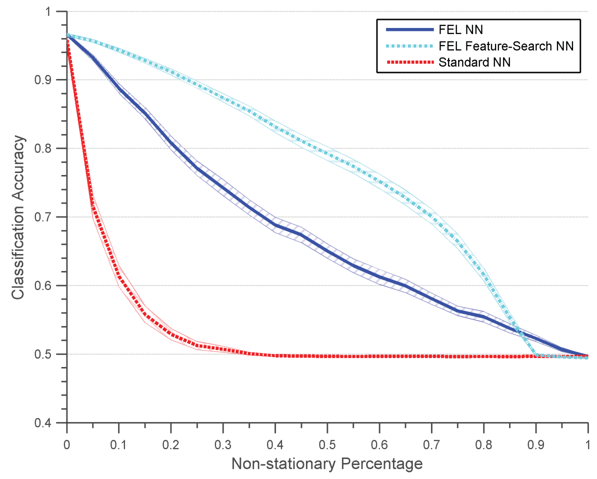

Fig. 10. Classification accuracy for the MNIST classification task when using four digits. Each line represents the mean classification accuracy, with the shaded proportion representing the 95% confidence interval. Nonstationary percentage refers to the percent of total training iterations that are performed using only two of the four possible digits.

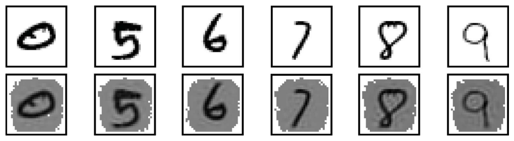

Fig. 11. Comparison of the original 1024-dimensional handwritten digits (top) and the PCA reduced and reconstructed 128-dimensional digits (bottom) for the new digits included in the full MNIST test.

present training examples of all four digits and 2) second phase—we only present training examples of two of the digits (digits one and two). The nonstationary percentage represents the proportion of training time spent in the second phase (where only two of the digits are used for training).

We evaluate the performance of a standard MLP network, a FEL network, and a FEL network using the feature-sign search (FEL-FS) algorithm. Each network has 128 inputs for the digit and four outputs representing the classification of that digit.

For both FEL networks, the number of hidden neurons is increased to 64, and the number of FEL neurons is increased to 512. We also increase the number of hidden neurons in the MLP network to 78. This number of hidden neurons is chosen to give the MLP network the same amount of learning resources as the FEL networks; each FEL network is able to change the weights between the 128 input neurons and the 64 hidden neurons as well as the weights between the 512 fixed expansion layer neurons and the four output neurons, giving the FEL networks 10,240 weights with which to store information. The MLP network is able to change the weights between the 128 input neurons and the 78 hidden neurons as well as the weights between the 78 hidden neurons and the four output neurons; this gives the MLP network 10,296 weights with which to store information.

The parameters for the MLP network and the FEL network are the same as those used in the single-learner Gaussian classification task. For the FEL-FS network, we use $\gamma = 3.5$.

TABLE V
CLASSIFICATION ACCURACY FOR THE MNIST CLASSIFICATION TASK WHEN USING FOUR DIGITS. *: DIFFERENCE BETWEEN THE VALUES IS NOT STATISTICALLY SIGNIFICANT

Non-stationary percentage	Standard MLP Network	FEL Network	**FEL Feature-Sign Network**
0.00	0.9595	0.9661	**0.9654**
0.25	0.5123	0.7708	**0.8930**
0.50	0.4967	0.6501	**0.7923**
0.75	0.4963	0.5629	**0.6643**
1.00	**0.4966***	**0.4964***	**0.4944***

TABLE VI
CLASSIFICATION ACCURACY FOR THE MNIST CLASSIFICATION TASK USING ALL 10 DIGITS. *: DIFFERENCE BETWEEN THE VALUES IS NOT STATISTICALLY SIGNIFICANT

Non-stationary percentage	Standard MLP Network	FEL Network	**FEL Feature-Sign Network**
0.00	0.8450	0.8609	**0.8735**
0.25	0.4805	0.6216	**0.6927**
0.50	0.4813	0.5429	**0.5765**
0.75	0.4812	**0.5002***	**0.4978***
1.00	0.4812	**0.4843***	**0.4845***

In addition, we do not apply the FEL weight initialization technique to the FEL-FS network (leaving the input-hidden weights randomly initialized as in the MLP network).

For each value of the nonstationary percentage, 100 independent test runs are performed for each algorithm, and the results used to determine the mean accuracy, standard deviation, and the 95% confidence interval for the mean accuracy.

A plot of the accuracy of the networks tested is shown in Fig. 10, with some detailed values being shown in Table V. These results clearly show that both the FEL and the FEL-FS networks perform significantly better than the standard MLP network on the more complex MNIST classification task, whereas the FEL-FS network consistently outperforms the basic FEL network. In addition, we can see that the performance of the FEL-FS network has a more linear degradation profile compared with the exponential degradation of the MLP and FEL networks.

E. Nonstationary MNIST Classification Task With 10 Digits

The previous task demonstrated MNIST classification using only the digits one, two, three, and four. The decision to use only four classes is made in an effort to present a more challenging classification task analogous to the earlier Gaussian cluster classification task (in that both tasks required classification into one of four possible classes). Of particular interest is how the FEL and FEL-FS networks perform when using all 10 of the digits within the MNIST dataset.

The previous nonstationary MNIST classification task is repeated using all of the digits (0–9) from the MNIST dataset. Reconstruction of PCA reduced samples of the additional digits are shown in Fig. 11. Aside from increasing the number of classes from four to ten, the same procedure, networks,

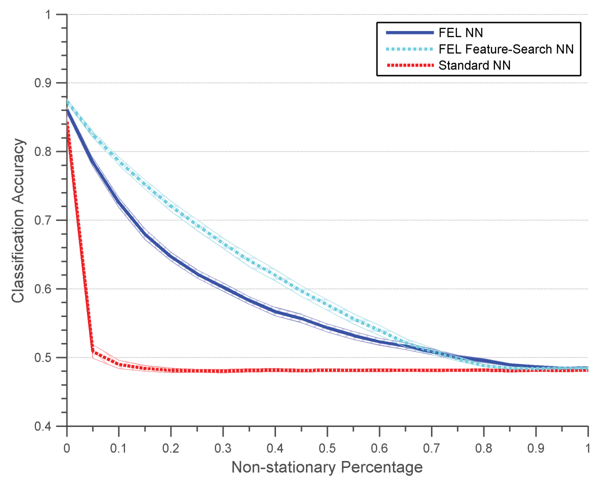

Fig. 12. Classification accuracy for the MNIST classification task when using all 10 digits. Each line represents the mean classification accuracy, with the shaded proportion representing the 95% confidence interval. Nonstationary percentage refers to the percent of total training iterations that are performed using only two of the four possible digits.

and parameters from the previous section are used for this test. During the first phase of training, we present training examples of all 10 digits. During the second phase of training, we only present training examples of five of the digits. The nonstationary percentage represents the proportion of training time spent in the second phase (where only five of the digits are presented).

For each value of the nonstationary percentage, 35 independent test runs are performed for each algorithm, along with the results used to determine the mean accuracy, standard deviation, and the 95% confidence interval for the mean accuracy.

The accuracy of the networks tested is shown in Fig. 12, with some detailed values shown in Table VI. The inclusion of all 10 digits renders this task much more challenging; accuracy decreased for all three methods. The relative performance of the FEL-FS, FEL, and MLP networks, however, remains consistent with that of the four digit classification task.

VII. Conclusion

This paper introduced the FEL neural network, which was uniquely designed to mitigate forgetting effects in parameterized supervised learning systems. This was achieved by exploiting sparse encoding for latching long-term representations. Learning was inherently achieved in an incremental, online manner, with modest requirements for additional memory and computational resources. In addition, the feature-sign search algorithm can be used to significantly improve FEL accuracy. When embedded in an ensemble of learners, the FEL exhibited significantly higher accuracy in the presence of nonstationary inputs, without the need for tuning any application-specific parameters. One of the interesting areas for further research was optimizing the confidence estimate to improve overall ensemble performance. Partitioning the training set such that each learner achieved higher accuracy in overlapping regions of the input space seemed to be another promising direction of investigation. The proposed framework can be directly applied to many other real-world machine learning tasks, such as time series prediction and anomaly detection.

VIII. Acknowledgment

The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusion 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 l’Académie Sci., Ser. III, Sci. Vie*, vol. 320, no. 12, pp. 989–997, 1997.
[2] B. Ans, S. Rousset, R. M. French, and S. Musca, “Self-refreshing memory in artificial neural networks: Learning temporal sequences without catastrophic forgetting,” *Connect. Sci.*, vol. 16, no. 2, pp. 71–99, 2004.
[3] L. Breiman, “Bagging predictors,” *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.
[4] G. Brown and J. Wyatt, “Negative correlation learning and the ambiguity family of ensemble methods,” in *Multiple Classifier Systems*. New York, NY, USA: Springer-Verlag, 2003, p. 161.
[5] R. Elwell and R. Polikar, “Incremental learning of concept drift in nonstationary environments,” *IEEE Trans. Neural Netw.*, vol. 22, no. 10, pp. 1517–1531, Oct. 2011.
[6] D. M. Endres and J. E. Schindelin, “A new metric for probability distributions,” *IEEE Trans. Inf. Theory*, vol. 49, no. 7, pp. 1858–1860, Jul. 2003.
[7] M. Frean and A. Robins, “Catastrophic forgetting in simple networks: An analysis of the pseudorehearsal solution,” *Netw., Comput. Neural Syst.*, vol. 10, no. 3, pp. 227–236, 1999.
[8] R. M. French, “Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks,” *Connect. Sci.*, vol. 4, nos. 3–4, pp. 365–378, 1992.
[9] R. M. French, “Catastrophic interference in connectionist networks: Can it be predicted, can it be prevented?” in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 1994, p. 1176.
[10] R. M. French, “Using pseudo-recurrent connectionist networks to solve the problem of sequential learning,” in *Proc. 19th Annu. Cognit. Sci. Soc. Conf.*, 1997, pp. 1–6.
[11] R. M. French and A. Ferrara, “Modeling time perception in rats: Evidence for catastrophic interference in animal learning,” in *Proc. 21st Annu. Conf. Cognit. Sci. Conf.*, 1999, pp. 173–178.
[12] M. Hattori, “Dual-network memory model using a chaotic neural network,” in *Proc. Int. Joint. Conf. Neural Netw.*, 2010, pp. 1–5.
[13] G. E. Hinton and D. C. Plaut, “Using fast weights to deblur old memories,” in *Proc. 9th Annu. Conf. Cognit. Sci. Soc.*, 1987, pp. 177–186.
[14] Y. Lecun and C. Cortes. (1998). *The MNIST Database of Handwritten Digits* [Online]. Available: http://yann.lecun.com/exdb/mnist/
[15] H. Lee, A. Battle, R. Raina, and A. Y. Ng, “Efficient sparse coding algorithms,” in *Advances in Neural Information Processing Systems*, vol. 19. Cambridge, MA, USA: MIT Press, 2007, p. 801.
[16] H. Lejeune, A. Ferrara, F. Simons, and J.H. Wearden, “Adjusting to changes in the time of reinforcement: Peak interval transitions in rats,” *J. Experim. Psychol., Animal Behavior Process.*, vol. 23, no. 2, pp. 211–231, 1997.
[17] J. P. Levy and D. Bairaktaris, “Connectionist dual-weight architectures,” *Lang. Cognit. Process.*, vol. 10, nos. 3–4, pp. 265–283, 1995.
[18] J. Lin, “Divergence measures based on the shannon entropy,” *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 145–151, Jan. 1991.
[19] M. McCloskey and N. J. Cohen, “Catastrophic interference in connectionist networks: The sequential learning problem,” *Psychol. Learn. Motivat.*, vol. 24, pp. 109–165, Jan. 1989.

[20] O.-M. Moe-Helgesen and H. Stranden, "Catastophic forgetting in neural networks," Dept. Comput. & Information Sci., Norwegian Univ. Science & Technology (NTNU), Trondheim, Norway, Tech. Rep., 2005.

[21] M. D. Muhlbaier, A. Topalis, and R. Polikar, "Learn^{++} .NC: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 152–168, Jan. 2009.

[22] I. Mukherjee, C. Rudin, and R. Schapire, "The rate of convergence of AdaBoost," in *Proc. 24th Annu. COLT*, 2011, pp. 537–558.

[23] N. C. Oza, R. Polikar, J. Kittler, and F. Roli, "Multiple classifier systems," in *Proc. 6th Int. Workshop MCS*, Seaside, CA, USA, Jun. 2005.

[24] R. Ratcliff, "Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions," *Psychol. Rev.*, vol. 97, no. 2, pp. 285–308, 1990.

[25] A. Robins, "Catastrophic forgetting in neural networks: The role of rehearsal mechanisms," in *Proc. 1st New Zealand Int. Two-Stream Conf. Artif. Neural Netw. Expert Syst.*, 1993, pp. 65–68.

[26] A. Robins, "Catastrophic forgetting, rehearsal and pseudorehearsal," *Connect. Sci.*, vol. 7, no. 2, pp. 123–146, 1995.

[27] S. Wang, K. Tang, and X. Yao, "Diversity exploration and negative correlation learning on imbalanced data sets," in *Proc. Int. Joint Conf. Neural Netw.*, Jun. 2009, pp. 3259–3266.

[28] S. Wang and X. Yao, "The effectiveness of a new negative correlation learning algorithm for classification ensembles," in *Proc. IEEE Int. Conf. Data Mining Workshops*, Dec. 2010, pp. 1013–1020.

Robert Coop (S'07) received the B.S., M.S., and Ph.D. degrees in computer science from the University of Tennessee, Knoxville, TN, USA, in 2008, 2011, and 2013, respectively.

He is currently working in the area of intelligent inference algorithms for risk assessment modeling. His current research interests include robust neural networks learning algorithms with particular focus on dealing with real-world and nonstationary data.

Aaron Mishtal (S'12) received the B.S. degree in computer science from the University of Tennessee, Knoxville, TN, USA, in 2012, where he is currently pursuing the Ph.D. degree with the Machine Intelligence Laboratory.

His current research interests include artificial general intelligence with a focus on scalable perception and control algorithms and architectures.

Itamar Arel (S'92–M'03–SM'06) received the B.S., M.S., and Ph.D. degrees in electrical and computer engineering in 1995, 1998, and 2003, respectively, and the M.B.A. degree from Ben-Gurion University, Beersheba, Israel.

He is currently an Associate Professor with the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN, USA. From 2000 to 2003, he was with TeraCross, Inc., Ramat Gan, Israel, where he was a Chief Scientist developing Terabit/sec switch fabric integrated circuits. His current research interests include the intersection of neural networks based deep machine learning and decision making under uncertainty. Dr. Arel is a recipient of the U.S. Department of Energy Early CAREER Award.