Digital Signal Processing ••• (••••) ••••••

Contents lists available at ScienceDirect

Digital Signal Processing

www.elsevier.com/locate/dsp

Efficient online learning with improved LSTM neural networks ☆

Ali H. Mirza [a], Mine Kerpicci [b,*], Suleyman S. Kozat [b]

[a] Ericsson, Stockholm, Sweden
[b] Department of Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Online learning
LSTM
Weight matrix factorization

ABSTRACT

We introduce efficient online learning algorithms based on the Long Short Term Memory (LSTM) networks that employ the covariance information. In particular, we introduce the covariance of the present and one-time step past input vectors into the gating structure of the LSTM networks. Additionally, we include the covariance of the output vector, and we learn their weight matrices to improve the learning performance of the LSTM networks where we also provide their updates. We reduce the number of system parameters through the weight matrix factorization where we convert the LSTM weight matrices into two smaller matrices in order to achieve high learning performance with low computational complexity. Moreover, we apply the introduced approach to the Gated Recurrent Unit (GRU) architecture. In our experiments, we illustrate significant performance improvements achieved by our methods on real-life datasets with respect to the vanilla LSTM and vanilla GRU networks.

1. Introduction

1.1. Preliminaries

In the modern era, online learning is getting more important and extensively studied in the literature due to its wide range of application areas including machine learning, signal processing and network intrusion detection [1–3]. In this paper, we study online learning for sequentially observed data in a nonlinear regression problem. We aim to sequentially learn the parameters of a system in time with low computational complexity.

In the computational learning literature, linear modeling approaches are used in various applications due to their low computational complexity [4]. However, such approaches are not suitable for the modeling of real life signals since they are not usually capable of capturing complex nonlinear relationships in the signal [1]. Therefore, nonlinear approaches are used in wide range of applications due to their high modeling power [5–8]. Among the nonlinear approaches, Deep Neural Networks (DNNs) are extensively used [9,10]. Nevertheless, DNN based methods suffer from limited performance on modeling the temporal data due to their lack of memory [11]. In order to mitigate this issue, Recurrent Neural Networks (RNNs), which are able to handle the long-term data dependency, are introduced [12], [13]. These methods, on the other hand, suffer from exploding and vanishing gradient problems during the back propagation of the error gradient [14]. The Long Short Term Memory (LSTM) networks solve this problem thanks to their gating architectures that control the flow of information [15]. Here, we introduce efficient online learning algorithms using the LSTM networks. Furthermore, since the gating mechanism of the LSTM network is its backbone, we incorporate additional information into the gating mechanism that helps in learning and modeling the complex processes effectively while preserving computational complexity [15].

There are many applications such as genetic protein therapy and housing prices where there is direct or indirect link between the past and the present data instances [16]. In such cases, using this relation, which is grasped with the covariance information, increases the prediction performance in an efficient manner. We utilize this covariance information in our LSTM networks through the gating structures. Furthermore, we also learn the covariance weight matrices during the training process. Hence, we improve the learning performance of the LSTM networks by not only utilizing covariance information but also learning its relation with the desired output. We call this improved LSTM networks based on the covariance information as “Co-LSTM” networks.

Although including the covariance information into the gating structure of the LSTM networks increases the modeling power, and hence, the learning performance, it also introduces additional time and computational complexity. This increase in time and computational complexity results from the additional input and output

☆ This work was supported by the Scientific and Technological Research Council of Turkey under Contract 117E153.

* Corresponding author.
E-mail addresses: ali.h.mirza@ericsson.com (A.H. Mirza), mine@ee.bilkent.edu.tr (M. Kerpicci), kozat@ee.bilkent.edu.tr (S.S. Kozat).

https://doi.org/10.1016/j.dsp.2020.102742
1051-2004/

weight and covariance matrices of the Co-LSTM networks. In order to make the resulting network to have low complexity, we incorporate the Weight Matrix Factorization (WMF) method into the Co-LSTM network weight matrices [17]. The objective of the WMF is to break down a higher rank matrix into two lower rank matrices. Hence, we significantly reduce the time and computational complexities with the use of the WMF on the Co-LSTM network weight matrices while maintaining the overall accuracy and the performance of the system.

1.2. Prior art and comparisons

In the learning literature, neural network based approaches are extensively used due to their high capabilities on modeling data [18]. Particularly, Long Short Term Memory (LSTM) networks provide significantly high learning performance thanks to their ability to learn the long-term data dependencies with their gating based architectures [19,15]. However, as a result of their performance-complexity trade-off, LSTM networks are not fully utilized in nonlinear regression tasks. In this paper, we solve nonlinear regression problem with our novel LSTM networks that bypass this trade-off. With our improved structure with WMF, we achieve higher learning performance and lower training time compared to the vanilla LSTM where we illustrate these significant improvements through our experiments.

Covariance matrix provides useful information on the data, which makes it highly valuable for learning tasks. However, its use in neural network based methods is highly limited due to their black box nature. The most common approach in the literature to utilize the covariance matrix in the neural networks is to use it for preprocessing such as dimensionality reduction [20,21]. Nonetheless, such preprocessing steps may degrade the learning performance due to information loss. In this paper, we directly employ the covariance matrix along with the data without any information loss in our novel LSTM structure, which results in high learning performance.

Matrix factorization method is highly used in neural network based algorithms for compression purposes [22,23]. These algorithms decrease the training time by reducing the number of training parameters that are naturally introduced with the basic neural network architecture [22]. However, they provide low performance since the direct application of this method degrades the learning capacity. In this paper, we apply the matrix factorization to our improved Co-LSTM structure, which provides high learning performance. Hence, even though we observe a slight degradation through WMF, we still achieve significantly high performance.

1.3. Contributions

The main contributions of this paper are as follows:

- We propose effective online learning algorithms based on the LSTM networks where we incorporate input and output covariance information, which grasp the relationship between the previous and present data instances, into the gating structure.
- We assign a weight matrix to each covariance matrix in our improved LSTM based gating structure and learn all the weights through the sequential updates where we enhance the regression performance.
- In order to alleviate the time and computational complexity, we use the Weight Matrix Factorization (WMF) method where we elegantly control trade off between computational complexity and performance.
- Through our experiments on real-life datasets, we illustrate significant performance improvements of the introduced Co-LSTM and Co-GRU networks over the vanilla LSTM and vanilla GRU networks, respectively.

1.4. Organization of the paper

This paper is organized as follows. In Section 2, we first describe our problem setting and the model formulation. We then introduce our models in Section 3.1 and provide the parameter updates in Section 3.2. In Section 3.3, we describe the Weight Matrix Factorization (WMF) method used to reduce the computational complexity. We provide the updates for the reformulation of our method based on the WMF in Section 3.4. In Section 4, we demonstrate the performance improvements of our methods on real datasets. Finally, we conclude with our remarks in Section 5.

2. Problem description

In this paper, all vectors are column vectors, and they are denoted by boldface lower case letters. Matrices are represented by boldface uppercase letters. For a vector $\boldsymbol{u}$, $\boldsymbol{u}^T$ is the ordinary transpose. For a vector $\boldsymbol{x}_t$, $x_{t,i}$ is the ith element of the vector $\boldsymbol{x}_t$. Similarly, for a matrix $\boldsymbol{W}$, w_{ij} is the ith row and the jth column entry.

We sequentially observe $\boldsymbol{x}_t \in \mathbb{R}^p$ along with desired output d_t at each discrete time or step t, and we aim to estimate the output as $\hat{d}_t$. Here, $\hat{d}_t$ depends only on $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t\}$ and $\{d_1, d_2, \ldots, d_{t-1}\}$ where we evaluate our loss function as

$$\ell_t(d_t, \hat{d}_t) = (d_t - \hat{d}_t)^2.$$

For this, we use the basic architecture of the RNNs [12], [13], which is defined as follows:

$$\boldsymbol{h}_t = \gamma(\boldsymbol{W}^{(h)}\boldsymbol{x}_t + \boldsymbol{R}^{(h)}\boldsymbol{h}_{t-1})$$
$$\boldsymbol{y}_t = \beta(\boldsymbol{R}^{(y)}\boldsymbol{h}_t),$$

where $\boldsymbol{h}_t \in \mathbb{R}^m$ is the state vector, $\boldsymbol{x}_t \in \mathbb{R}^p$ is the input vector and $\boldsymbol{y}_t \in \mathbb{R}^m$ is the output vector of the RNN. Here, the functions $\gamma(\cdot)$ and $\beta(\cdot)$ apply pointwise to the vectors and are commonly set to $\tanh(\cdot)$. Furthermore, $\boldsymbol{W}^{(h)} \in \mathbb{R}^{m\times p}$, $\boldsymbol{R}^{(h)} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{R}^{(y)} \in \mathbb{R}^m$ are the coefficient weight matrices.

In our formulation, we use the LSTM networks as a special case of the RNNs in order to calculate the estimate of the desired output, i.e., $\hat{d}_t$. The LSTM network with one hidden layer is defined as follows [15]:

$$\tilde{\boldsymbol{c}}_t = \tanh\left(\boldsymbol{W}^{(\tilde{c})}\boldsymbol{x}_t + \boldsymbol{R}^{(\tilde{c})}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(\tilde{c})}\right) \quad (1)$$
$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}^{(i)}\boldsymbol{x}_t + \boldsymbol{R}^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)}\right) \quad (2)$$
$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}^{(f)}\boldsymbol{x}_t + \boldsymbol{R}^{(f)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(f)}\right) \quad (3)$$
$$\boldsymbol{c}_t = \boldsymbol{\Pi}_t^{(\boldsymbol{i})}\tilde{\boldsymbol{c}}_t + \boldsymbol{\Pi}_t^{(\boldsymbol{f})}\boldsymbol{c}_{t-1} \quad (4)$$
$$\boldsymbol{o}_t = \sigma\left(\boldsymbol{W}^{(o)}\boldsymbol{x}_t + \boldsymbol{R}^{(o)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(o)}\right) \quad (5)$$
$$\boldsymbol{h}_t = \boldsymbol{\Pi}_t^{(\boldsymbol{o})}\tanh(\boldsymbol{c}_t), \quad (6)$$

where $\boldsymbol{c}_t \in \mathbb{R}^m$ is the state vector, $\boldsymbol{x}_t \in \mathbb{R}^p$ is the input vector and $\boldsymbol{h}_t \in \mathbb{R}^m$ is the output vector. $\boldsymbol{W}^{(\cdot)} \in \mathbb{R}^{m\times p}$ and $\boldsymbol{R}^{(\cdot)} \in \mathbb{R}^{m\times m}$ are the weight matrices and $\boldsymbol{b}^{(\cdot)} \in \mathbb{R}^m$ is the bias vector. Here, $\boldsymbol{i}_t$, $\boldsymbol{f}_t$ and $\boldsymbol{o}_t$ are the input, forget and output gates, respectively. Also, $\boldsymbol{\Pi}_t^{(\boldsymbol{i})} = \mathrm{diag}(\boldsymbol{i}_t)$, $\boldsymbol{\Pi}_t^{(\boldsymbol{f})} = \mathrm{diag}(\boldsymbol{f}_t)$ and $\boldsymbol{\Pi}_t^{(\boldsymbol{o})} = \mathrm{diag}(\boldsymbol{o}_t)$ where $\mathrm{diag}(\cdot)$ is the diagonal matrix. The sigmoid function $\sigma(\cdot)$ and $\tanh(\cdot)$ applies pointwise to the vector elements. Then, the final estimate of the desired output is

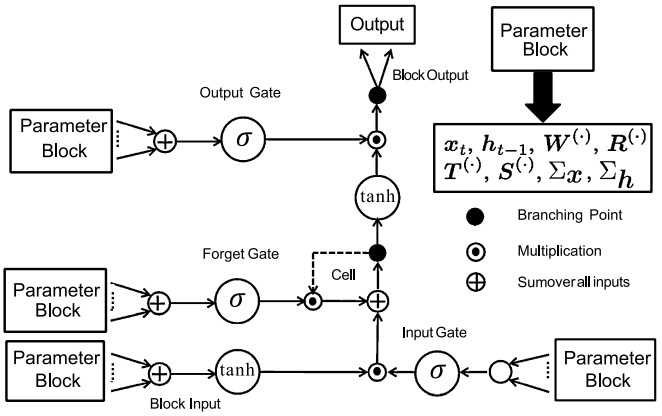

Fig. 1. The diagram of the Co-LSTM networks. Here, we have covariance information parameters $\boldsymbol{T}^{(\cdot)}$, $\boldsymbol{S}^{(\cdot)}$, $\Sigma_{\boldsymbol{x}}$ and $\Sigma_{\boldsymbol{h}}$ in addition to $\boldsymbol{x}_t$, $\boldsymbol{h}_{t-1}$ and vanilla LSTM network parameters such as $\boldsymbol{W}^{(\cdot)}$ and $\boldsymbol{R}^{(\cdot)}$.

$$\hat{d}_t = \boldsymbol{p}_t^T \boldsymbol{h}_t, \tag{7}$$

where $\boldsymbol{p}_t \in \mathbb{R}^m$ is the final regression coefficients. Here, we also learn $\boldsymbol{p}_t$ from the data in an online manner.

Note that our aim corresponds to choosing the parameters of the constructed system, which are represented with a set of equations, such that the total loss $\sum_{t=1}^{T} l_t(d_t, \hat{d}_t)$ is minimized. For this, we update the parameters iteratively (with the Stochastic Gradient Descent (SGD) algorithm [24]) based on each newly observed data instance to optimize the specified objective function. To minimize the total loss, we also consider improving the structure of the existing LSTM model by incorporating additional information, which provides better modeling of the system. Hence, we include new system parameters, which are learned through sequential updates along with the existing LSTM parameters.

3. Efficient online learning algorithms with covariance information and WMF

In this section, we present our LSTM models based on the weighted input and output covariances. Then, we apply the weight matrix factorization method to our proposed models to achieve low computational complexity. Finally, we provide the stochastic gradient based weight updates for the introduced method.

3.1. Covariance matrix based LSTM networks (Co-LSTM)

We first define the covariance matrix of $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t-1}$ as $\Sigma_{\boldsymbol{x}} = \text{Cov}(\boldsymbol{x}_t, \boldsymbol{x}_{t-1})$ where $\Sigma_{\boldsymbol{x}} \in \mathbb{R}^{p \times p}$. Then, we introduce this covariance information into the gating structure of the original LSTM networks [6,19] where we propose three different models for its use. We provide the basic diagram of the proposed Co-LSTM networks in Fig. 1. The introduced network structure takes covariance matrices as an input and learns the corresponding weight matrices from the data in addition to the parameters of the vanilla LSTM architecture as shown in Fig. 1. Note that we only provide the variations applied to the LSTM network for the input gate, while the variations for the remaining gates are done in a similar manner.

To this end, in Section 3.1.1, we use both input and output covariance matrices in addition to the existing parameters of the LSTM network to achieve better learning performance. Moreover, we use input covariance matrices in Section 3.1.2 to modify the gate equations without introducing new weight matrices to obtain this performance with low computational complexity. Finally, in Section 3.1.3, we additionally include output covariance matrices to increase the learning performance even further.

3.1.1. Model 1: additive weighted input and output covariance information model

In the first model, we include both input and output covariance information in the gating structure of the vanilla LSTM network. We redefine the gate equations by assigning different weights to the covariance matrices where we learn them in a sequential manner. Here, the input gate is represented as

$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}^{(i)}\boldsymbol{x}_t + \boldsymbol{R}^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)} + \boldsymbol{T}^{(i)}\Sigma_{\boldsymbol{x}}\boldsymbol{x}_t + \boldsymbol{S}^{(i)}\Sigma_{\boldsymbol{h}}\boldsymbol{h}_{t-1}\right), \tag{8}$$

where $\boldsymbol{T}^{(i)} \in \mathbb{R}^{m \times p}$ and $\boldsymbol{S}^{(i)} \in \mathbb{R}^{m \times m}$ are the weight matrices for the covariance matrices of the input and output vectors, respectively. In (8), $\Sigma_{\boldsymbol{x}} \in \mathbb{R}^{p \times p}$ is the input covariance matrix, i.e., $\Sigma_{\boldsymbol{x}} = \text{Cov}(\boldsymbol{x}_t, \boldsymbol{x}_{t-1})$, and $\Sigma_{\boldsymbol{h}} \in \mathbb{R}^{m \times m}$ is the output covariance matrix, i.e., $\Sigma_{\boldsymbol{h}} = \text{Cov}(\boldsymbol{h}_t, \boldsymbol{h}_{t-1})$. We can obtain the remaining LSTM equations for the forget and the output gates by following a similar structure as in (8).

3.1.2. Model 2: weighted input covariance model

In the second model, we modify the vanilla LSTM equations to include the input covariance information. We use the input vectors in multiplication with the input covariance matrices such that we do not introduce any additional weight matrix. Then, the input gate is written as

$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}^{(i)}\Sigma_{\boldsymbol{x}}\boldsymbol{x}_t + \boldsymbol{R}^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)}\right). \tag{9}$$

3.1.3. Model 3: weighted input and output covariance model

In the third model, we modify the weighted input and the output vectors in the vanilla LSTM network by introducing the weighted covariances as

$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}^{(i)}\Sigma_{\boldsymbol{x}}\boldsymbol{x}_t + \boldsymbol{R}^{(i)}\Sigma_{\boldsymbol{h}}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)}\right). \tag{10}$$

3.2. Gradient based update calculations for the Co-LSTM networks

In this subsection, we derive the parameter updates of the introduced method based on the Stochastic Gradient Descent (SGD) algorithm [24,25]. We define our loss function as $\ell_t(d_t, \hat{d}_t) = (d_t - \hat{d}_t)^2$. Then, we have the following SGD update for the weight vector

$$\begin{aligned}\boldsymbol{p}_{t+1} &= \boldsymbol{p}_t - \mu\nabla_{\boldsymbol{p}_t}\ell_t(d_t, \hat{d}_t)\\ &= \boldsymbol{p}_t + 2\mu(d_t - \hat{d}_t)\boldsymbol{h}_t.\end{aligned}$$

For the parameter α, which is defined as an element of any weight matrix of the Co-LSTM structure, we have

$$\begin{aligned}\alpha_{t+1} &= \alpha_t - \mu\frac{\partial\ell_t(d_t, \hat{d}_t)}{\partial\alpha}\\ &= \alpha_t + 2\mu(d_t - \hat{d}_t)\boldsymbol{p}^T\frac{\partial\boldsymbol{h}_t}{\partial\alpha},\end{aligned}$$

where μ is the learning rate. Here, $\frac{\partial\boldsymbol{h}_t}{\partial\alpha}$ is calculated as follows

$$\frac{\partial\boldsymbol{h}_t}{\partial\alpha} = \boldsymbol{\Pi}_t^{(\boldsymbol{o}')}\tanh(\boldsymbol{c}_t) + \boldsymbol{\Pi}_t^{(\boldsymbol{o})}\boldsymbol{\Pi}_t^{(\tanh'(\boldsymbol{c}_t))}\frac{\partial\boldsymbol{c}_t}{\partial\alpha}, \tag{11}$$

where we define $\boldsymbol{u}' = \frac{\partial\boldsymbol{u}}{\partial\alpha}$ for a vector $\boldsymbol{u}$. Then, $\frac{\partial\boldsymbol{c}_t}{\partial\alpha}$ and $\boldsymbol{o}' = \frac{\partial\boldsymbol{o}_t}{\partial\alpha}$ are written as follows

$$\frac{\partial\boldsymbol{c}_t}{\partial\alpha} = \boldsymbol{\Pi}_t^{(\boldsymbol{i}')}\tilde{\boldsymbol{c}}_t + \boldsymbol{\Pi}_t^{(\boldsymbol{i})}\boldsymbol{\Pi}_t^{(\tanh'(\boldsymbol{\beta}))}\frac{\partial\boldsymbol{\beta}}{\partial\alpha} + \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{f}')}\boldsymbol{c}_{t-1} + \boldsymbol{\Pi}_t^{(\boldsymbol{f})}\boldsymbol{\gamma}_{c_{t-1}}^{(\alpha)}, \tag{12}$$

$$\frac{\partial \boldsymbol{o}_t}{\partial \alpha} = \boldsymbol{\Pi}_t^{(\sigma'(\boldsymbol{\rho}))} \frac{\partial \boldsymbol{\rho}}{\partial \alpha}, \tag{13}$$

where

$$\boldsymbol{\beta} = \boldsymbol{W}^{(\tilde{c})}\boldsymbol{x}_t + \boldsymbol{R}^{(\tilde{c})}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(\tilde{c})} + \boldsymbol{T}^{(\tilde{c})}\Sigma_{\boldsymbol{x}}\boldsymbol{x}_t + \boldsymbol{S}^{(\tilde{c})}\Sigma_{\boldsymbol{h}}\boldsymbol{h}_{t-1},$$
$$\boldsymbol{\rho} = \boldsymbol{W}^{(o)}\boldsymbol{x}_t + \boldsymbol{R}^{(o)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(o)} + \boldsymbol{T}^{(o)}\Sigma_{\boldsymbol{x}}\boldsymbol{x}_t + \boldsymbol{S}^{(o)}\Sigma_{\boldsymbol{h}}\boldsymbol{h}_{t-1},$$

and

$$\boldsymbol{\gamma}_{c_{t-1}}^{(\alpha)} = \frac{\partial \boldsymbol{c}_{t-1}}{\partial \alpha}.$$

Particularly, for $\alpha = w_{ij}^{(i)}$, the gradient based updates as stated in (11), (12) and (13) can be written as follows

$$\frac{\partial \boldsymbol{h}_t}{\partial w_{ij}^{(i)}} = \boldsymbol{\Pi}_t^{(\boldsymbol{o}')} \tanh(\boldsymbol{c}_t) + \boldsymbol{\Pi}_t^{(\boldsymbol{o})} \boldsymbol{\Pi}_t^{(\tanh'(\boldsymbol{c}_t))} \frac{\partial \boldsymbol{c}_t}{\partial w_{ij}^{(i)}}, \tag{14}$$

where $\boldsymbol{\Pi}_t^{(\boldsymbol{o}')} = \operatorname{diag}\left(\frac{\partial \boldsymbol{o}_t}{\partial w_{ij}^{(i)}}\right)$. Then, $\frac{\partial \boldsymbol{o}_t}{\partial w_{ij}^{(i)}}$ and $\frac{\partial \boldsymbol{c}_t}{\partial w_{ij}^{(i)}}$ can be explicitly written as

$$\begin{aligned}\frac{\partial \boldsymbol{o}_t}{\partial w_{ij}^{(i)}} = \boldsymbol{\Pi}_t^{(\sigma'(\boldsymbol{\rho}))} \Big(& \boldsymbol{R}^{(o)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o}')} \tanh(\boldsymbol{c}_{t-1}) \\ & + \boldsymbol{R}^{(o)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o})} \boldsymbol{\Pi}_{t-1}^{(\tanh'(\boldsymbol{c}))} \boldsymbol{\gamma}_{ij,t-1}^{(i)} \\ & + \boldsymbol{A}'^{(o)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o})} \tanh(\boldsymbol{c}_{t-1}) + \boldsymbol{A}^{(o)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o}')} \tanh(\boldsymbol{c}_{t-1}) \\ & + \boldsymbol{A}^{(o)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o})} \boldsymbol{\Pi}_{t-1}^{(\tanh'(\boldsymbol{c}))} \boldsymbol{\gamma}_{ij,t-1}^{(i)} \Big)\end{aligned}$$

and

$$\frac{\partial \boldsymbol{c}_t}{\partial w_{ij}^{(i)}} = \boldsymbol{\Pi}_t^{(\tilde{\boldsymbol{c}})} \frac{\partial \boldsymbol{i}_t}{\partial w_{ij}^{(i)}} + \boldsymbol{\Pi}_t^{(\boldsymbol{i})} \frac{\partial \tilde{\boldsymbol{c}}_t}{\partial w_{ij}^{(i)}} + \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{c})} \frac{\partial \boldsymbol{f}_t}{\partial w_{ij}^{(i)}} + \boldsymbol{\Pi}_t^{(\boldsymbol{f})} \boldsymbol{\gamma}_{ij,t-1}^{(i)}, \tag{15}$$

where $\boldsymbol{A}^{(\cdot)} = \boldsymbol{S}^{(\cdot)}\Sigma_{\boldsymbol{h}}$ and

$$\begin{aligned}\frac{\partial \boldsymbol{i}_t}{\partial w_{ij}^{(i)}} = \boldsymbol{\Pi}_t^{(\sigma'(\boldsymbol{\theta}))} \Big(& \boldsymbol{\Delta}_{ij}\boldsymbol{x}_t + \boldsymbol{R}^{(i)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o}')} \tanh(\boldsymbol{c}_{t-1}) \\ & + \boldsymbol{R}^{(i)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o})} \boldsymbol{\Pi}_{t-1}^{(\tanh'(\boldsymbol{c}))} \boldsymbol{\gamma}_{ij,t-1}^{(i)} \\ & + \boldsymbol{A}'^{(i)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o})} \tanh(\boldsymbol{c}_{t-1}) + \boldsymbol{A}^{(i)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o}')} \tanh(\boldsymbol{c}_{t-1}) \\ & + \boldsymbol{A}^{(i)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o})} \boldsymbol{\Pi}_{t-1}^{(\tanh'(\boldsymbol{c}))} \boldsymbol{\gamma}_{ij,t-1}^{(i)} \Big),\end{aligned} \tag{16}$$

where $\boldsymbol{\Delta}_{ij}$ is a matrix of zeros except 1 at the ith row and the jth column. Here, the rest of the updates for the parameters of other input gate weight matrices can be calculated similar to (16) by taking $\boldsymbol{\Delta}_{ij} = \boldsymbol{0}$ with $\boldsymbol{\theta} = \boldsymbol{W}^{(i)}\boldsymbol{x}_t + \boldsymbol{R}^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)} + \boldsymbol{T}^{(i)}\Sigma_{\boldsymbol{x}}\boldsymbol{x}_t + \boldsymbol{S}^{(i)}\Sigma_{\boldsymbol{h}}\boldsymbol{h}_{t-1}$ and changing the derivatives accordingly.

3.3. Weight matrix factorization based Co-LSTM networks (WMF-Co-LSTM)

In this subsection, we introduce the WMF based Co-LSTM networks (WMF-Co-LSTM). We employ the approach used in [17,26] where we decompose a matrix into two smaller matrices to reduce the total number of trainable parameters, which is increased by the introduction of covariance information to the gating structure of the LSTM networks. For a matrix $\boldsymbol{D} \in \mathbb{R}^{m\times p}$, we write $\boldsymbol{D} \approx \boldsymbol{D}_1\boldsymbol{D}_2$ where $\boldsymbol{D}_1 \in \mathbb{R}^{m\times q}$ and $\boldsymbol{D}_2 \in \mathbb{R}^{q\times p}$ such that

$q << \min(m, p)$.

Table 1
Total number of parameters of the vanilla LSTM, Co-LSTM and WMF-Co-LSTM Networks. Here, $q << \min(m, p)$ and $n << m$.

Algorithms	Number of parameters
Vanilla LSTM	$4(pm + mm + m)$
Co-LSTM (Model 1)	$4(2(pm + mm) + m)$
Co-LSTM (Model 2-3)	$4(pm + mm + m)$
WMF-Co-LSTM (Model 1)	$4(2(pq + qm + 2mn) + m)$
WMF-Co-LSTM (Model 2-3)	$4(pq + qm + 2mn + m)$

We apply the WMF approach to the weight matrices of all the models of the Co-LSTM networks, i.e., $\boldsymbol{W}^{(\cdot)} \approx \boldsymbol{W}_1^{(\cdot)}\boldsymbol{W}_2^{(\cdot)}$, $\boldsymbol{R}^{(\cdot)} \approx \boldsymbol{R}_1^{(\cdot)}\boldsymbol{R}_2^{(\cdot)}$, $\boldsymbol{S}^{(\cdot)} \approx \boldsymbol{S}_1^{(\cdot)}\boldsymbol{S}_2^{(\cdot)}$ and $\boldsymbol{T}^{(\cdot)} \approx \boldsymbol{T}_1^{(\cdot)}\boldsymbol{T}_2^{(\cdot)}$.

In Table 1, we provide the total number of parameters of the vanilla LSTM networks without WMF and Co-LSTM networks with and without WMF. Clearly, the total number of parameters for the Co-LSTM networks (specifically Model 1) increases by two-fold as compared to the total number of parameters for the vanilla LSTM networks. Hence, in order to alleviate the total number of parameters for the Co-LSTM networks, we use the WMF approach, which drastically reduces the number of parameters while still maintaining the learning performance as shown in our experiments.

Remark. The choice of q and n for WMF is based on $q << \min(m, p)$ and $n << m$. Under a reasonable assumption of $n = q$, we notice that the number of parameters of Co-LSTM and their corresponding WMF-Co-LSTM models become equal if $q = \frac{m(m+p)}{3m+p}$. Note that this corresponds to choosing q such that $\frac{m}{3} < q < m$, and hence the number of WMF-Co-LSTM parameters are always smaller than the number of Co-LSTM parameters under our assumptions.

3.4. Gradient based update calculations for WMF-Co-LSTM networks

In this subsection, we provide the gradient based calculations for the Co-LSTM networks (specifically Model 1) with the WMF. For instance, the input gate of the Model 1 of the Co-LSTM networks can be written as

$$\begin{aligned}\boldsymbol{i}_t = \sigma \Big(& \boldsymbol{W}_1^{(i)}\boldsymbol{W}_2^{(i)}\boldsymbol{x}_t + \boldsymbol{R}_1^{(i)}\boldsymbol{R}_2^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)} \\ & + \boldsymbol{T}_1^{(i)}\boldsymbol{T}_2^{(i)}\Sigma_{\boldsymbol{x}}\boldsymbol{x}_t + \boldsymbol{S}_1^{(i)}\boldsymbol{S}_2^{(i)}\Sigma_{\boldsymbol{h}}\boldsymbol{h}_{t-1} \Big).\end{aligned} \tag{17}$$

Rest of the gating equations can be written similarly by following (17). By following the procedure in Section 3.2, for $\alpha = w_{1_{ij}}^{(i)}$, where $w_{1_{ij}}^{(i)}$ is the ith row and the jth column entry of $\boldsymbol{W}_1^{(i)}$, the gradient based updates as stated in (11), (12) and (13) can be written as follows

$$\frac{\partial \boldsymbol{h}_t}{\partial w_{1_{ij}}^{(i)}} = \boldsymbol{\Pi}_t^{(\boldsymbol{o}')} \tanh(\boldsymbol{c}_t) + \boldsymbol{\Pi}_t^{(\boldsymbol{o})} \boldsymbol{\Pi}_t^{(\tanh'(\boldsymbol{c}_t))} \frac{\partial \boldsymbol{c}_t}{\partial w_{1_{ij}}^{(i)}}, \tag{18}$$

where $\boldsymbol{\Pi}_t^{(\boldsymbol{o}')} = \operatorname{diag}\left(\frac{\partial \boldsymbol{o}_t}{\partial w_{1_{ij}}^{(i)}}\right)$ and $\frac{\partial \boldsymbol{o}_t}{\partial w_{1_{ij}}^{(i)}}$, $\frac{\partial \boldsymbol{c}_t}{\partial w_{1_{ij}}^{(i)}}$ can be explicitly written as

$$\begin{aligned}\frac{\partial \boldsymbol{o}_t}{\partial w_{1_{ij}}^{(i)}} = \boldsymbol{\Pi}_t^{(\sigma'(\boldsymbol{\rho}))} \Big(& \boldsymbol{R}_1^{(o)}\boldsymbol{R}_2^{(o)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o}')} \tanh(\boldsymbol{c}_{t-1}) \\ & + \boldsymbol{R}_1^{(o)}\boldsymbol{R}_2^{(o)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o})} \boldsymbol{\Pi}_{t-1}^{(\tanh'(\boldsymbol{c}))} \boldsymbol{\gamma}_{ij,t-1}^{(i)} \\ & + \boldsymbol{\Theta}'^{(o)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o})} \tanh(\boldsymbol{c}_{t-1}) + \boldsymbol{\Theta}^{(o)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o}')} \tanh(\boldsymbol{c}_{t-1}) \\ & + \boldsymbol{\Theta}^{(o)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o})} \boldsymbol{\Pi}_{t-1}^{(\tanh'(\boldsymbol{c}))} \boldsymbol{\gamma}_{ij,t-1}^{(i)} \Big)\end{aligned} \tag{19}$$

Table 2
Steady state error performances of the algorithms on Kinematics dataset.

q \ Algorithms	Vanilla LSTM	Co-LSTM (Model 1)	Co-LSTM (Model 2)	Co-LSTM (Model 3)	WMF-Co-LSTM (Model 1)	WMF-Co-LSTM (Model 2)	WMF-Co-LSTM (Model 3)
2	0.4866	0.4460	0.4959	0.2017	0.4620	0.4913	0.2120
3					0.4618	0.4888	0.2011
4					0.4615	0.4845	0.2008

and

$$\frac{\partial \boldsymbol{c}_t}{\partial w_{1_{ij}}^{(i)}} = \boldsymbol{\Pi}_t^{(\tilde{\boldsymbol{c}})} \frac{\partial \boldsymbol{i}_t}{\partial w_{1_{ij}}^{(i)}} + \boldsymbol{\Pi}_t^{(\boldsymbol{i})} \frac{\partial \tilde{\boldsymbol{c}}_t}{\partial w_{1_{ij}}^{(i)}} + \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{c})} \frac{\partial \boldsymbol{f}_t}{\partial w_{1_{ij}}^{(i)}} + \boldsymbol{\Pi}_t^{(\boldsymbol{f})} \boldsymbol{\gamma}_{ij,t-1}^{(i)},$$

where $\boldsymbol{\Theta}^{(\cdot)} = \boldsymbol{S}_1^{(\cdot)} \boldsymbol{S}_2^{(\cdot)} \Sigma_{\boldsymbol{h}}$ and

$$\begin{aligned}\frac{\partial \boldsymbol{i}_t}{\partial w_{1_{ij}}^{(i)}} = \boldsymbol{\Pi}_t^{(\sigma'(\boldsymbol{\theta}))} \Bigg(& \boldsymbol{\Pi}_t^{(\boldsymbol{W}_{2_j})} \boldsymbol{x}_t + \boldsymbol{R}_1^{(i)} \boldsymbol{R}_2^{(i)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o}')} \tanh(\boldsymbol{c}_{t-1}) \\ & + \boldsymbol{R}_1^{(i)} \boldsymbol{R}_2^{(i)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o})} \boldsymbol{\Pi}_{t-1}^{(\tanh'(\boldsymbol{c}))} \boldsymbol{\gamma}_{ij,t-1}^{(i)} + \boldsymbol{\Theta}'^{(i)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o})} \tanh(\boldsymbol{c}_{t-1}) \\ & + \boldsymbol{\Theta}^{(i)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o}')} \tanh(\boldsymbol{c}_{t-1}) + \boldsymbol{\Theta}^{(i)} \boldsymbol{\Pi}_{t-1}^{(\boldsymbol{o})} \boldsymbol{\Pi}_{t-1}^{(\tanh'(\boldsymbol{c}))} \boldsymbol{\gamma}_{ij,t-1}^{(i)} \Bigg), \end{aligned} \tag{20}$$

where $\boldsymbol{W}_{2_j}$ is the jth row of the matrix $\boldsymbol{W}_2$. Then, the rest of the gradients are similar to (20) with $\boldsymbol{W}_{2_j} = \boldsymbol{0}$ and $\boldsymbol{\theta} = \boldsymbol{W}_1^{(i)} \boldsymbol{W}_2^{(i)} \boldsymbol{x}_t + \boldsymbol{R}_1^{(i)} \boldsymbol{R}_2^{(i)} \boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)} + \boldsymbol{T}_1^{(i)} \boldsymbol{T}_2^{(i)} \Sigma_{\boldsymbol{x}} \boldsymbol{x}_t + \boldsymbol{S}_1^{(i)} \boldsymbol{S}_2^{(i)} \Sigma_{\boldsymbol{h}} \boldsymbol{h}_{t-1}$. Hence, this concludes our derivations of the training procedure for the Co-LSTM with WMF using the SGD.

4. Experiments

In this section, we illustrate the regression performance of the proposed models on real-life datasets including Kinematics [27], Elevators [28], Protein Tertiary [29] and Puma8NH [30]. We demonstrate the classification performance of the proposed models on two real-life dynamic datasets, i.e., ISCX IDS 2012 [31] and Pen-Based Handwritten Digits [32] dataset. Moreover, we compare the proposed models with the state-of-the-art methods based on their performances on image caption generation [33] and sentiment analysis [34] by using the MS COCO dataset [35] and IMDB movie review dataset [36], respectively. Note that all of these datasets with various behaviors and structures are particularly chosen to show that using the relation between the past and present data points through covariance matrices can be highly useful in many real life applications. Below we provide a brief description of each dataset used in the experiments.

- **Kinematics:** This dataset contains the data belonging to the real-life simulation of the 8-link robot arm. The realistic simulation carried out in this dataset is based on the forward dynamics of the 8-link robot arm. Our aim here is to predict the distance of the robot arm to the target, effectively. This dataset consists of 8192 data samples with the dimension of $\boldsymbol{x}_t \in \mathbb{R}^5$.
- **Elevators:** Our aim on this dataset is to predict the variables related to the movement of the F16 aircraft. In this data set, we have 8192 data samples with the dimension of $\boldsymbol{x}_t \in \mathbb{R}^{18}$.
- **Protein Tertiary:** This dataset consists of data points, which are carefully collected through the Critical Assessment of protein Structure Prediction (CASP) experiments for protein structure prediction. This study captures the inter-relation and dependency of various features of the proteins. The number of data samples and the input dimension for this dataset is 16559 and $\boldsymbol{x}_t \in \mathbb{R}^9$, respectively.
- **Puma8NH:** This dataset is a realistic modeling of the dynamics of the Unimation Puma 560 robotic arm where we predict the angular acceleration of one of the arms of the robot using various important features such as angular position, velocity and torque of the robot arm. The input dimension of this dataset is $\boldsymbol{x}_t \in \mathbb{R}^9$ with a total of 45730 data samples.
- **ISCX IDS 2012:** The ISCX IDS dataset consists of the network payloads captured for seven days for FTP, SMTP, HTTP, SSH and POP3 traffic formats where the aim is to detect the anomalies in the network payloads. The anomalies in this dataset originates from a diverse set of multi-stage network attacks. Each network payload contains two components namely; source port and destination port. The payloads are in a hexadecimal format where each character is of 6 bits thus constituting a vocabulary of 64.
- **Pen-Based Handwritten Digits Recognition:** This dataset contains approximately 11000 handwriting samples collected from 44 writers each contributing 250 samples. These samples are collected on a 500×500 pixel tablet. The input dimension for this data set is $\boldsymbol{x}_t \in \mathbb{R}^{16}$.
- **MS COCO:** The Microsoft Common Objects in Context (MS COCO) dataset contains 123,287 images, i.e., a diversified collection of images containing various daily life objects.
- **IMDB Movie Review:** This dataset is constructed with 50000 reviews obtained from IMDB for sentiment analysis. The dataset contains 25000 reviews for training and 25000 reviews for testing.

4.1. Regression and classification tasks

In this section, we illustrate the performances of the proposed Co-LSTM models and vanilla LSTM network for the regression and classification problems. We also incorporate our new approach with covariance matrices and WMF to the well known GRU architecture and provide performance comparisons for all architectures.

Note that, in all experiments, we perform 5-fold cross validation and specify the parameters such that the algorithms achieve their best performances. Furthermore, we perform an experiment on the Kinematics dataset [27] to choose q for WMF. Our aim on this dataset is to predict the distance of the end-effector from a target based on the five-dimensional input vectors belonging to the robotic arm. We provide the regression performances of the WMF-Co-LSTM, i.e., Co-LSTM with WMF, models for various q values in addition the vanilla LSTM and Co-LSTM models in Table 2. We observe that the performance of the WMF-Co-LSTM models slightly increases as q increases. More importantly, they provide even better results than their corresponding models without WMF in some cases since they also remove the overfitting issues. However, increasing q also increases the training time. Since our aim is to achieve high performance while also decreasing the computational time, we specify $q = 2$ for the rest of our experiments.

We provide the regression performances of all Co-LSTM network models and vanilla LSTM network on the Kinematics dataset [27] in Fig. 2a where we choose the learning rate as $\mu = 0.01$. Through our experiments, we observe that as we add the additional covariance information to the gating structure of the vanilla LSTM networks, the overall performance naturally increases. Clearly, this covariance information provides better learning par-

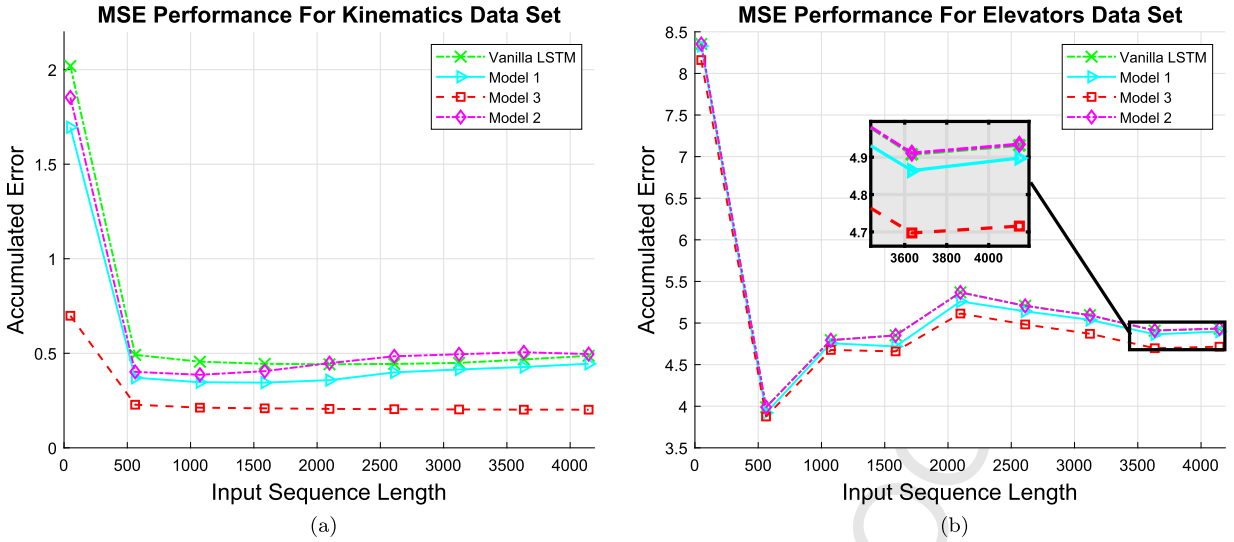

Fig. 2. Accumulated errors for the distance prediction performance of (a) the robotic arm on the Kinematics dataset (b) an F16 aircraft on the Elevators dataset using various Co-LSTM network models and vanilla LSTM networks (without the WMF).

Table 3
The total number of the parameters for the vanilla LSTM and Co-LSTM (all models) networks with and without WMF for Kinematics, Elevators, Protein Tertiary and Puma8NH Datasets.

Algorithms \ Datasets	Kinematics	Elevators	Protein	Puma8NH
Vanilla LSTM	220	2664	684	684
Co-LSTM (Model 1)	420	5256	1332	1332
Co-LSTM (Model 2)	220	2664	684	684
Co-LSTM (Model 3)	220	2664	684	684
WMF-Co-LSTM (Model 1)	340	1224	612	612
WMF-Co-LSTM (Model 2)	**180**	**648**	**324**	**324**
WMF-Co-LSTM (Model 3)	**180**	**648**	**324**	**324**

Table 4
The training time (in seconds), which is obtained with a computer with i5-6400 processor, 2.7 GHz CPU and 16 GB RAM, for the vanilla LSTM and all the models of the Co-LSTM networks with and without WMF.

Algorithms \ Datasets	Kinematics	Elevators	Protein	Puma8NH
Vanilla LSTM	63.27	768.14	430.08	450.18
Co-LSTM (Model 1)	139.14	1762.12	870.73	907.01
Co-LSTM (Model 2)	68.06	770.10	450.68	463.63
Co-LSTM (Model 3)	71.19	780.97	473.33	480.17
WMF-Co-LSTM (Model 1)	115.19	370.77	390.36	391.08
WMF-Co-LSTM (Model 2)	62.20	191.84	190.36	187.24
WMF-Co-LSTM (Model 3)	65.19	196.69	191.57	188.88

Table 5
Steady state error performances of all models of Co-LSTM networks with and without WMF and various state-of-the-art deep learning algorithms.

Algorithms \ Datasets	Kinematics	Elevators	Protein	Puma8NH
RNN	0.6921	5.8255	0.3334	0.1984
Bidirectional RNN	0.6103	5.4449	0.3101	0.1800
Composite LSTM	0.5239	4.8522	0.2460	0.1162
ResNet-152	0.5019	4.9996	0.2684	0.1316
AlexNet	0.5572	5.9211	0.3100	0.1662
VGG-16	0.6994	6.0021	0.3562	0.2000
Vanilla LSTM	0.4866	4.930	0.2569	0.1284
Bidirectional LSTM	0.4120	4.8886	0.2314	0.1222
Co-LSTM (Model 1)	0.4460	4.898	0.2381	0.1111
Co-LSTM (Model 2)	0.4959	4.935	0.2679	0.1287
Co-LSTM (Model 3)	**0.2017**	**4.716**	**0.1764**	**0.1079**
WMF-Co-LSTM (Model 1)	0.4620	4.990	0.2578	0.1207
WMF-Co-LSTM (Model 2)	0.4913	4.942	0.2699	0.1301
WMF-Co-LSTM (Model 3)	**0.2120**	**4.721**	**0.1780**	**0.1086**

ticularly when the data instances are highly correlated, and hence increases the overall regression performance. As shown in Fig. 2a, Model 3 outperforms all the other models of the Co-LSTM networks and vanilla LSTM network since its structure incorporates both input and output covariance matrices.

In Fig. 2b, we obtain the regression performances of all algorithms on the Elevators dataset [28] where we specify the learning rate as $\mu = 0.01$. In this case, we have 18-dimensional input vector, and based on this, we aim to predict the set of actions done by the F16 aircraft. We observe in Fig. 2b that Model 3 of the Co-LSTM networks has the best performance among all algorithms with the use of input and output covariance matrices. Similarly, Model 1 has the second best regression performance while Model 2 and vanilla LSTM network provide almost same accumulated error trend. Since the only model that do not use the output covariance is Model 2, we conclude that the output covariance matrix provides as useful information as the input covariance in the learning process from the data.

We list the number of parameters and provide the training time (in seconds) for the vanilla LSTM and Co-LSTM networks with and without the WMF for all datasets, i.e., Kinematics [27], Elevators [28], Protein Tertiary [29] and Puma8NH [30], in Table 3 and Table 4, respectively. Note that we use a computer with i5-6400 processor, 2.7 GHz CPU and 16 GB RAM. Clearly, the number of parameters for the Co-LSTM Model 1 is greater than the vanilla LSTM having equal number of parameters with Co-LSTM Model 2 and Model 3. We emphasize that particularly with our choice of $q = 2$, WMF-Co-LSTM 2 and Model 3 have significantly less number of parameters than the vanilla LSTM as seen in Table 3. These results are also supported by the training time of the algorithms provided in Table 4. Model 1 of the Co-LSTM networks without WMF needs almost twice more time for training as compared to the vanilla LSTM networks. However, we significantly reduce the training time with the use of the WMF as shown in Table 4. As expected, Model 2 and Model 3 have the smallest training time among all algorithms due to their low number of parameters. However, we observe that the training time of Model 3 is slightly greater than Model 2 since it incorporates both input and output covariance matrices while Model 2 is using only the input covariance.

Moreover, we provide the steady state error performances of the algorithms to highlight the performance improvement of our introduced models for all datasets, i.e., Kinematics [27], Elevators [28], Protein Tertiary [29] and Puma8NH [30], in Table 5. Here, we

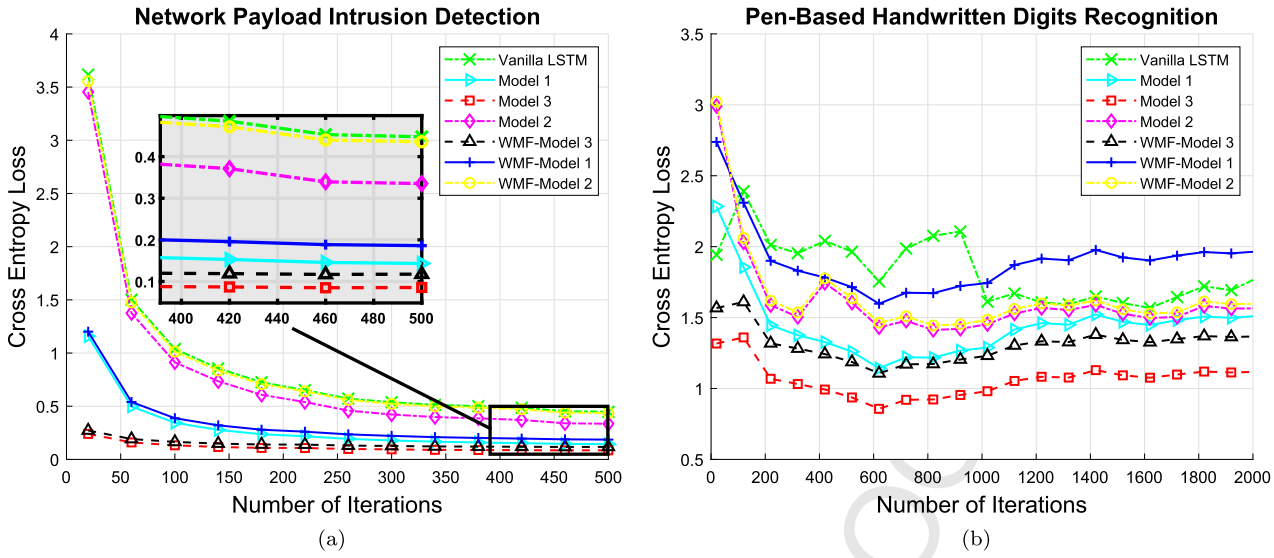

Fig. 3. Cross entropy losses of vanilla LSTM and all models of Co-LSTM networks with and without WMF for (a) network payload intrusion detection (b) Pen-Based Handwritten Digits classification.

compare the performance of our models with the vanilla LSTM, RNNs [12] [13], Bidirectional RNNs [37], Composite LSTM [38] and Bidirectional LSTM networks [39] along with the state-of-the-art deep learning models, i.e., AlexNet [40], VGG-16 [41] and ResNet-152 [42]. As seen in Table 5, Bidirectional RNNs/LSTMs perform better than the vanilla RNN/LSTM networks as expected due to their improved processing models. We observe that Model 3 of our proposed method provides significantly better regression performance than the state-of-the-art deep learning models as well as composite LSTMs and Bidirectional RNNs/LSTMs due to the use of input and output covariance. Furthermore, the deep learning models, i.e., AlexNet, VGG-16 and ResNet-152, contain significantly higher number of trainable parameters, which results in higher training time compared to the proposed Co-LSTM networks. We also observe that Model 1 performs better than the compared methods but it may result in overfitting issues due to large number of parameters. Therefore, even though it also incorporates input and output covariance, its performance is worse than Model 3. Finally, Model 2 provides similar performance to the vanilla LSTM since it only employs the input covariance. However, it also decreases the computational time with WMF, which still makes it a better choice than vanilla LSTM.

We perform an experiment on ISCX IDS dataset [31] to demonstrate the anomaly detection performance of the algorithms where we use both the source and destination network payloads to capture the anomalies. Here, the payloads are in a hexadecimal format where each character is of 6 bits thus constituting a vocabulary of 64. We use one hot encoding to convert these characters into 64-dimensional feature vectors. For the experiments, we specify the optimal networks parameters such that $\mu = 0.001$ and $q = 32$. In Fig. 3a, we observe that all models the Co-LSTM networks outperforms the vanilla LSTM network. Among the introduced models, Model 3 provides the best performance since it utilize both input and output data relations through covariance matrices, which provide highly useful information for such a dynamic process.

Moreover, we demonstrate the classification performances of the vanilla LSTM and all the models of Co-LSTM networks on the Pen-Based Handwritten Digits Recognition dataset [32]. This dataset contains handwritten digits (from 0 to 9) obtained by collecting 250 samples from 44 writers. The input dimension for this data set is $\boldsymbol{x}_t \in \mathbb{R}^{16}$ where we use the learning rate $\mu = 0.001$ and rank $q = 2$. We provide the cross entropy losses of the algorithms in Fig. 3b. In this case, we observe a different trend such that Model 1 and Model 2 provide lower performance than the Vanilla LSTM at first since the partial use of covariance information might be misleading for such dataset involving classes with similar behaviors. However, even Model 2 converges to a better performance than the Vanilla LSTM in time with better learning of the model through covariance matrices. Moreover, Model 3 of the Co-LSTM networks has the best performance during the whole process since it separately incorporates both input and output covariance matrices, which elegantly deal with the data variations.

Finally, we consider the regression performances of the algorithms based on the GRU networks. Since our approach is generic, we apply our method, which is mainly proposed for the LSTM architecture, to the GRU architecture, which is described by the following set of equations:

$$\tilde{\boldsymbol{z}}_t = \sigma(\boldsymbol{W}^{(\tilde{z})}\boldsymbol{x}_t + \boldsymbol{R}^{(\tilde{z})}\boldsymbol{y}_{t-1}) \tag{21}$$

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}^{(r)}\boldsymbol{x}_t + \boldsymbol{R}^{(r)}\boldsymbol{y}_{t-1}) \tag{22}$$

$$\tilde{\boldsymbol{y}}_t = g(\boldsymbol{W}^{(y)}\boldsymbol{x}_t + \boldsymbol{r}_t \odot (\boldsymbol{R}^{(y)}\boldsymbol{y}_{t-1})) \tag{23}$$

$$\boldsymbol{y}_t = \tilde{\boldsymbol{y}}_t \odot \tilde{\boldsymbol{z}}_t + \boldsymbol{y}_{t-1} \odot (\boldsymbol{1} - \tilde{\boldsymbol{z}}_t), \tag{24}$$

where $\boldsymbol{x}_t \in \mathbb{R}^p$ is the input vector and $\boldsymbol{y}_t \in \mathbb{R}^m$ is the output vector. The functions $g(\cdot)$ and $\sigma(\cdot)$ are set to the hyperbolic tangent and sigmoid functions, respectively. For the coefficient matrices, we have $\boldsymbol{W}^{(\tilde{z})} \in \mathbb{R}^{m\times p}$, $\boldsymbol{R}^{(\tilde{z})} \in \mathbb{R}^{m\times m}$, $\boldsymbol{W}^{(z)} \in \mathbb{R}^{m\times p}$, $\boldsymbol{R}^{(z)} \in \mathbb{R}^{m\times m}$, $\boldsymbol{W}^{(z)} \in \mathbb{R}^{m\times p}$ and $\boldsymbol{R}^{(z)} \in \mathbb{R}^{m\times m}$. Here, $\tilde{\boldsymbol{z}}_t$ and $\boldsymbol{r}_t$ are the update and the reset gates, respectively. To obtain GRU-based algorithms, we directly replace the LSTM equations with the GRU equations and apply our proposed procedures. Note that the GRU networks lack the output gate, which controls the amount of the incoming memory content. Furthermore, these networks differ in the location of the forget gates and the corresponding reset gates.

We perform similar set of experiments (as conducted with the LSTM networks) by using the GRU networks and provide the performance results, i.e., steady state error performances in Table 6 and the training time (in seconds) in Table 7. Based on these results, we observe a similar trend to the LSTM based models such that Model 3 provides the best regression performance among all algorithms.

Please cite this article in press as: A.H. Mirza et al., Efficient online learning with improved LSTM neural networks, Digit. Signal Process. (2020), https://doi.org/10.1016/j.dsp.2020.102742

Table 6
Steady state error performances of vanilla GRU and all models of Co-GRU networks with and without WMF.

Algorithms \ Datasets	Kinematics	Elevators	Protein	Puma8NH
Vanilla GRU	0.4712	0.4899	0.2303	0.1284
Co-GRU (Model 1)	0.4555	4.8978	0.2283	0.1113
Co-GRU (Model 2)	0.4951	4.9280	0.2470	0.1282
Co-GRU (Model 3)	**0.2100**	**4.7321**	**0.1999**	**0.1083**
WMF-Co-GRU (Model 1)	0.4613	4.993	0.2423	0.1201
WMF-Co-GRU (Model 2)	0.4963	4.9420	0.2502	0.1311
WMF-Co-GRU (Model 3)	**0.2166**	**4.7499**	**0.2015**	**0.1095**

Table 7
The training time (in seconds), which is obtained with a computer with i5-6400 processor, 2.7 GHz CPU and 16 GB RAM, for the vanilla GRU and all models of the Co-GRU networks with and without WMF.

Algorithms \ Datasets	Kinematics	Elevators	Protein	Puma8NH
Vanilla GRU	51.87	666.61	390.98	410.10
Co-GRU (Model 1)	112.64	154.99	780.43	867.21
Co-GRU (Model 2)	58.01	690.13	423.28	411.11
Co-GRU (Model 3)	62.29	690.27	423.47	400.12
WMF-Co-GRU (Model 1)	105.10	310.12	340.22	337.79
WMF-Co-GRU (Model 2)	52.20	161.34	120.46	157.24
WMF-Co-GRU (Model 3)	55.44	126.49	111.56	140.18

Table 8
The performance results of all Co-LSTM networks and the compared methods with the values in percentage (%) based on the metrics of BLEU-4, METEOR and ROUGE-L.

Algorithms \ Metrics	BLEU-4	METEOR	ROUGE-L
Up-Down	36.20	27.00	56.40
RFNet	35.80	27.40	56.80
GCN-LSTM	36.80	27.90	57.00
NIC	29.60	25.20	52.60
SC-Tanh	30.00	24.88	51.50
Vanilla LSTM	29.60	25.20	52.60
Co-LSTM (Model 1)	36.90	28.20	57.19
Co-LSTM (Model 2)	36.60	27.77	56.80
Co-LSTM (Model 3)	**37.03**	**28.23**	**57.24**
WMF-Co-LSTM (Model 1)	36.30	26.67	56.82
WMF-Co-LSTM (Model 2)	36.24	25.90	56.12
WMF-Co-LSTM (Model 3)	37.01	27.11	56.99

Table 9
Sentiment classification accuracies in percentage (%) and AUC scores of all Co-LSTM networks and the compared methods.

Algorithms \ Metrics	Positive sentiments accuracy	Negative sentiments accuracy	AUC ccore
Single Layer Feedforward NN	84.21	86.11	0.8749
Vanilla LSTM	89.82	90.01	0.9212
Co-LSTM (Model 1)	92.05	91.99	0.9357
Co-LSTM (Model 2)	91.87	91.66	0.9301
Co-LSTM (Model 3)	92.11	92.57	0.9487
WMF-Co-LSTM (Model 1)	90.09	90.67	0.9219
WMF-Co-LSTM (Model 2)	90.00	90.03	0.9214
WMF-Co-LSTM (Model 3)	90.15	89.99	0.9226

4.2. Image caption generation task

In this section, we compare the performances of our proposed models with the state-of-the-art methods including vanilla LSTM, Up-Down [43], RFNet [44], GCN-LSTM [45], Show and Tell NIC [46] and end-to-end gLSTM networks SC-Tanh [47] on the problem of image caption generation. For this, we use the MS COCO dataset [35], which contains 123287 images of daily objects in natural context along with their five different descriptions.

We provide the results based on various metrics such as BLEU [48], METEOR [49] and ROUGE-L [50] in Table 8. We observe that Model 3 of our proposed Co-LSTM networks outperforms the other methods in all of the compared metrics. The reason is that the covariance matrices provide highly valuable information about the observed data, which results in better and more detailed learning than the other approaches. We also provide some examples of the captions generated by utilizing vanilla LSTM and our Co-LSTM Model 3 method along with the two of the ground truth descriptions in Fig. 4. We observe that our model is able to learn more detailed information from the image, and hence its image descriptions are significantly closer to the ground truths.

Table 10
Sentiment classification accuracies in percentage (%) of WMF-Co-LSTM networks and the compared methods.

Algorithms	Overall accuracy
Vanilla LSTM	89.92
LSTMs w2v	90.40
Hierarchical Attention	92.00
Skim-LSTM	91.20
Paragraph Vector	92.70
Paragraph Vector (2-layer MLP)	94.50
LSTM LM + BLSTM	92.35
cBLSTM LM + BLSTM	92.83
WMF-Co-LSTM (Model 1)	90.38
WMF-Co-LSTM (Model 2)	90.02
WMF-Co-LSTM (Model 3)	90.07

4.3. Sentiment analysis task

In this section, we evaluate the performances of our models on the binary sentiment analysis task. For this, we use the IMDB movie review dataset [36], which contains highly polar movie reviews. In our experiments, the maximum length of a review is 500 while the average review length is approximately 234 words. Hence, we apply zero padding for the reviews shorter than 500 to equate the sequence lengths.

We compare the performances of our Co-LSTM network models with the single layer feedforward neural network and the vanilla LSTM. In Table 9, we provide the positive and negative sentiment classification accuracies with the AUC scores. As seen from the results, all of our models outperform both the single layer feedforward neural network and the vanilla LSTM. Particularly, Model 3 of our Co-LSTM networks achieve the highest classification performance with its high learning capability through covariance matrices.

Moreover, we include the performance results of the pretrained word2vec LSTM model [51] [52], Hierarchical Attention Networks [53], Skim-LSTM networks [54], Paragraph Vector [55], Paragraph Vector (2-layer MLP) [56], LSTM LM + BLSTM and cBLSTM LM + BLSTM binary classifier [57] as in Table 10. Note that our proposed methods have lower complexity and hence require lower training time than the compared methods. Moreover, in Table 10, we observe that the proposed algorithms (particularly Model 2) provide higher performance than the compared state-of-the-art algorithms due to their high modeling capabilities.

5. Conclusion

We have introduced efficient online learning algorithms based on the LSTM network architecture. We have utilized the input and the output covariance information and introduced them into the gating structure of the vanilla LSTM networks. We have proposed three variants of the introduced Co-LSTM network approach where we assign weight matrices to the input and output covariance matrices and learn them in a sequential manner. Furthermore, we have used the matrix factorization method on the Co-LSTM network weight matrices to reduce the time and computational complexity while maintaining the error performances.

Please cite this article in press as: A.H. Mirza et al., Efficient online learning with improved LSTM neural networks, Digit. Signal Process. (2020), https://doi.org/10.1016/j.dsp.2020.102742

LSTM: A guy is doing a trick on a bike.

Co-LSTM (Model 3): A mountain biker jumps a rock on a mountain.

GT1: A mountain biker is riding on a line back at bushes.

GT2: A biker jumps and doing tricks in a forest.

LSTM: A man is riding a surfboard.

Co-LSTM (Model 3): A man rides a wave on a surfboard.

GT1: A man rides a surfboard and makes a splash.

GT2: A man rides a surfboard as a wave makes a splash.

LSTM: A man rides a bicycle on a trail down a river.

Co-LSTM (Model 3): A man is riding a bicycle on a trail through some trees.

GT1: A man rides a mountain bike down a slope in the woods.

GT2: A man rides a bicycle on a trail down a river.

Fig. 4. The examples of captions generated by the vanilla LSTM and Co-LSTM network (Model 3) with two corresponding ground truths, i.e., GT1, GT2.

In the experiments, we compare the training time and the total number of parameters of the vanilla LSTM network and Co-LSTM networks with and without the WMF to demonstrate the trade-offs of our introduced methods. Consequently, we demonstrate significant performance improvements achieved by the introduced algorithms with respect to the state-of-the-art methods on several real datasets.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A.C. Singer, G.W. Wornell, A.V. Oppenheim, Nonlinear autoregressive modeling and estimation in the presence of noise, Digit. Signal Process. 4 (4) (1994) 207–221.
[2] I. Ali, Y.-T. Chen, Design quality and robustness with neural networks, IEEE Trans. Neural Netw. 10 (6) (1999) 1518–1527.
[3] V. Vovk, Competitive on-line linear regression, in: Advances in Neural Information Processing Systems, 1998, pp. 364–370.
[4] D.C. Montgomery, E.A. Peck, G.G. Vining, Introduction to Linear Regression Analysis, vol. 821, John Wiley & Sons, 2012.
[5] N. Cesa-Bianchi, G. Lugosi, Prediction, Learning, and Games, Cambridge University Press, 2006.
[6] K. Greff, R.K. Srivastava, J. Koutník, B.R. Steunebrink, J. Schmidhuber, Lstm: a search space odyssey, IEEE Trans. Neural Netw. Learn. Syst. 28 (10) (2017) 2222–2232.
[7] M. Iliadis, L. Spinoulas, A.K. Katsaggelos, Deep fully-connected networks for video compressive sensing, Digit. Signal Process. 72 (2018) 9–18, https://doi.org/10.1016/j.dsp.2017.09.010.
[8] T.G. Kang, J.W. Shin, N.S. Kim, Dnn-based monaural speech enhancement with temporal and spectral variations equalization, Digit. Signal Process. 74 (2018) 102–110, https://doi.org/10.1016/j.dsp.2017.12.002.
[9] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Netw. 61 (2015) 85–117.
[10] G. Montavon, W. Samek, K.-R. Müller, Methods for interpreting and understanding deep neural networks, Digit. Signal Process. 73 (2018) 1–15, https://doi.org/10.1016/j.dsp.2017.10.011.
[11] M. Hermans, B. Schrauwen, Training and analysing deep recurrent neural networks, in: C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, vol. 26, Curran Associates, Inc., 2013, pp. 190–198.
[12] J. Mazumdar, R.G. Harley, Recurrent neural networks trained with backpropagation through time algorithm to estimate nonlinear load harmonic currents, IEEE Trans. Ind. Electron. 55 (9) (2008) 3484–3491, https://doi.org/10.1109/TIE.2008.925315.
[13] H. Jaeger, Tutorial on Training Recurrent Neural Networks, Covering BPPT, RTRL, EKF and the "Echo State Network" Approach, vol. 5, GMD-Forschungszentrum Informationstechnik, Bonn, 2002.
[14] Y. Bengio, P. Simard, P. Frasconi, et al., Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. 5 (2) (1994) 157–166.
[15] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780, https://doi.org/10.1162/neco.1997.9.8.1735.
[16] X. Liu, Deep recurrent neural network for protein function prediction from sequence, arXiv preprint, arXiv:1701.08318, 2017.
[17] O. Kuchaiev, B. Ginsburg, Factorization tricks for lstm networks, arXiv preprint, arXiv:1703.10722, 2017.
[18] D.F. Specht, A general regression neural network, IEEE Trans. Neural Netw. 2 (6) (1991) 568–576, https://doi.org/10.1109/72.97934.
[19] F.A. Gers, J. Schmidhuber, F. Cummins, Learning to Forget: Continual Prediction with Lstm, 1999.
[20] M.J. Er, S. Wu, J. Lu, H.L. Toh, Face recognition with radial basis function (rbf) neural networks, IEEE Trans. Neural Netw. 13 (3) (2002) 697–710.
[21] K. Hadad, M. Mortazavi, M. Mastali, A.A. Safavi, Enhanced neural network based fault detection of a vver nuclear power plant with the aid of principal component analysis, IEEE Trans. Nucl. Sci. 55 (6) (2008) 3611–3619, https://doi.org/10.1109/TNS.2008.2006491.
[22] T.N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, B. Ramabhadran, Low-rank matrix factorization for deep neural network training with high-dimensional output targets, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 6655–6659.

[23] Y. Zhang, E. Chuangsuwanich, J. Glass, Extracting deep neural network bottleneck features using low-rank matrix factorization, in: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2014, pp. 185–189.
[24] S. Ruder, An overview of gradient descent optimization algorithms, arXiv preprint, arXiv:1609.04747, 2016.
[25] R.J. Williams, D. Zipser, A learning algorithm for continually running fully recurrent neural networks, Neural Comput. 1 (2) (1989) 270–280.
[26] D.D. Lee, H.S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788.
[27] Delve data sets, http://www.cs.toronto.edu/~delve/data/datasets.html.
[28] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Mult.-Valued Log. Soft Comput. 17 (2011).
[29] M. Lichman, UCI Machine Learning Repository, 2013.
[30] L. Torgo, Regression data sets.
[31] A. Shiravi, H. Shiravi, M. Tavallaee, A.A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Comput. Secur. 31 (3) (2012) 357–374.
[32] D. Dua, C. Graff, UCI machine learning repository, http://archive.ics.uci.edu/ml, 2017.
[33] O. Vinyals, A. Toshev, S. Bengio, D. Erhan, Show and tell: a neural image caption generator, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3156–3164.
[34] Q.T. Ain, M. Ali, A. Riaz, A. Noureen, M. Kamran, B. Hayat, A. Rehman, Sentiment analysis using deep learning techniques: a review, Int. J. Adv. Comput. Sci. Appl. 8 (6) (2017) 424.
[35] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft coco: common objects in context, in: European Conference on Computer Vision, Springer, 2014, pp. 740–755.
[36] A.L. Maas, R.E. Daly, P.T. Pham, D. Huang, A.Y. Ng, C. Potts, Learning word vectors for sentiment analysis, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Portland, Oregon, USA, 2011, pp. 142–150.
[37] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (11) (1997) 2673–2681.
[38] N. Srivastava, E. Mansimov, R. Salakhudinov, Unsupervised learning of video representations using lstms, in: International Conference on Machine Learning, 2015, pp. 843–852.
[39] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional lstm and other neural network architectures, Neural Netw. 18 (5–6) (2005) 602–610.
[40] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[41] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint, arXiv:1409.1556, 2014.
[42] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[43] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, L. Zhang, Bottom-up and top-down attention for image captioning and visual question answering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6077–6086.
[44] W. Jiang, L. Ma, Y.-G. Jiang, W. Liu, T. Zhang, Recurrent fusion network for image captioning, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 499–515.
[45] T. Yao, Y. Pan, Y. Li, T. Mei, Exploring visual relationship for image captioning, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 684–699.
[46] O. Vinyals, A. Toshev, S. Bengio, D. Erhan, Show and tell: a neural image caption generator, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3156–3164.
[47] L. Zhou, C. Xu, P. Koch, J.J. Corso, Watch what you just said: image captioning with text-conditional attention, in: Proceedings of the on Thematic Workshops of ACM Multimedia 2017, 2017, pp. 305–313.
[48] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, Bleu: a method for automatic evaluation of machine translation, in: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, 2002, pp. 311–318.
[49] S. Banerjee, A. Lavie, Meteor: an automatic metric for mt evaluation with improved correlation with human judgments, in: Proceedings of the acl Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, 2005, pp. 65–72.
[50] C.-Y. Lin, Rouge: a package for automatic evaluation of summaries, in: Text Summarization Branches Out, 2004, pp. 74–81.
[51] X. Zhang, J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, in: Advances in Neural Information Processing Systems, 2015, pp. 649–657.
[52] D. Chai, W. Wu, Q. Han, F. Wu, J. Li, Description based text classification with reinforcement learning, arXiv preprint, arXiv:2002.03067, 2020.
[53] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, E. Hovy, Hierarchical attention networks for document classification, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 1480–1489.
[54] M. Seo, S. Min, A. Farhadi, H. Hajishirzi, Neural speed reading via skim-rnn, arXiv preprint, arXiv:1711.02085, 2017.
[55] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: International Conference on Machine Learning, 2014, pp. 1188–1196.
[56] J. Hong, M. Fang, Sentiment analysis with deeply learned distributed representations of variable length texts, Stanford University Report, 2015.
[57] A. Mousa, B. Schuller, Contextual bidirectional long short-term memory recurrent neural network language models: a generative approach to sentiment analysis, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, 2017, pp. 1023–1032.

Ali H. Mirza received the B.S. degree in with honors in Electrical Engineering from University of Engineering and Technology, Lahore, Pakistan, in 2014 and his M.S. degree in Electrical and Electronics Engineering from Bilkent University, Ankara, Turkey. He is currently working as a Developer for Stability in 4G/5G Systems at Ericsson HQ, Sweden. His research interests include machine learning, signal processing, online learning and deep neural networks.

Mine Kerpicci received the B.S. degree in Electrical and Electronics Engineering from Middle East Technical University, Ankara, Turkey in 2017. She is currently pursuing the M.S. degree in the Department of Electrical and Electronics Engineering at Bilkent University. Her research interests include machine learning, online learning and statistical signal processing.

Suleyman Serdar Kozat received the B.S. (Hons.) degree from Bilkent University, Ankara, Turkey, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana-Champaign, Urbana, IL, USA.

He joined the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, as a Research Staff Member and later became a Project Leader with Pervasive Speech Technologies Group, where he focused on problems related to statistical signal processing and machine learning. He was a Research Associate with the Cryptography and Anti-Piracy Group, Microsoft Research, Redmond, WA, USA. He is currently a Professor at the Electrical and Electronics Engineering Department at Bilkent University. He has co-authored over 120 papers in refereed high impact journals and conference proceedings and holds several patent inventions (currently used in several different Microsoft and IBM products, such as MSN and ViaVoice). He holds several patent inventions due to his research accomplishments with the IBM Thomas J. Watson Research Center and Microsoft Research. His current research interests include cyber security, anomaly detection, big data, data intelligence, adaptive filtering and machine learning algorithms for signal processing.

Dr. Kozat received many international and national awards. He is the Elected President of the IEEE Signal Processing Society, Turkey Chapter.

Please cite this article in press as: A.H. Mirza et al., Efficient online learning with improved LSTM neural networks, Digit. Signal Process. (2020), https://doi.org/10.1016/j.dsp.2020.102742