Online Additive Updates with FFT-IFFT operator on the GRU Neural Networks

Ali H. Mirza
Department of Electrical and Electronics Engineering
Bilkent University, Ankara 06800, Turkey
mirza@ee.bilkent.edu.tr

***Abstract*—In this paper, we derived the online additive updates of gated recurrent unit (GRU) network by using fast fourier transform-inverse fast fourier transform (FFT-IFFT) operator. In the gating process of the GRU networks, we work in the frequency domain and execute all the linear operations. For the non-linear functions in the gating process, we first shift back to the time domain and then apply non-linear GRU gating functions. Furthermore, in order to reduce the computational complexity and speed up the training process, we apply weight matrix factorization (WMF) on the FFT-IFFT variant GRU network. We then compute the online additive updates of the FFT-WMF based GRU networks using stochastic gradient descent (SGD) algorithm. We also used long short-term memory (LSTM) networks in place of the GRU networks. Through an extensive set of experiments, we illustrate that our proposed algorithm achieves a significant increase in performance with a decrease in computational complexity.**

Keywords. GRU, online learning, FFT, online updates, SGD

I. Introduction

Online learning is of vital importance and is widely studied in machine learning, adaptive signal processing and neural network literature [1], [2], [3]. In most of the online learning applications, nonlinear models are preferred over linear models because they are capable of modelling highly complex structures [4]. On the other hand, these nonlinear approaches suffer from overfitting problems, stability and convergence issues [5], [6]. Deep neural networks (DNNs) are highly used to model such nonlinear structures [7]. However, DNNs can provide only limited performance in modelling time series and processing temporal data [8]. As a result, recurrent neural networks (RNNs) are introduced, which not only handle the temporal data but also handle the time dependencies in the data as well [9], [10].

After the neural network structure is fixed, there exists a wide range of different methods to train the corresponding parameters in an online manner. Most commonly, the first-order gradient-based algorithms are used to train the parameters of the neural network. The Backpropagation Through Time (BPTT) algorithm is most commonly used and is highly efficient in computing gradients [10]. Training the parameters of the RNN is a difficult task and is highly computationally expensive [9]. For this purpose, the GRU and LSTM networks are used [11]. These networks have a gating mechanism that controls the flow of information and handles the long-term data dependency. For the parameters of the GRU network, we derive the SGD updates, i.e., also known as real-time recurrent learning (RTRL) algorithm, to minimize the instantaneous loss.

978-1-5386-1501-0/18/$31.00

In this paper, we process the linear operation gating part of the GRU network in the frequency domain. We convert the time sequential information into the frequency domain using discrete fourier transform (DFT) [12]. For the computational ease, we use FFT in our simulations to speed up the training process [13]. Once we process the linear part of the individual gates of the GRU network, we convert back to time domain using inverse discrete fourier transform (IDFT) and then apply the non-linear part of the gating function. These non-linear functions are commonly set to sigmoid or $\tanh$. Furthermore, we reduce the computational complexity of the proposed algorithm by using WMF [14]. We split the gating weight matrices of the GRU network into two smaller dimensions (rank) sub-matrices and then apply the FFT-IFFT operator on it. These smaller rank matrices essentially try to approximate the original weight matrix [15]. This decrease in the computational complexity is of supreme importance for the cases where we have a very large data set with huge dimensions. We then also derive the online additive updates for both the cases, i.e., the GRU network with and without the WMF. In section II, we describe the model and problem description and derive the online updates for the proposed algorithm. In section III, we illustrate the performance of the proposed algorithm through an extensive set of experiments on both the GRU and LSTM networks. In section IV, we provided the concluding remarks.

II. Problem Description

In this paper, all vectors are column vectors and denoted by boldface lower case letters. Matrices are represented by boldface upper case letters. For a vector $\boldsymbol{u}$, $|\boldsymbol{u}|$ is the $\ell 1$-norm and $\boldsymbol{u}^T$ is the ordinary transpose. For a matrix $\boldsymbol{W}$, w_{ij} is the i^{th} row and j^{th} column entry.

For the regression framework, we receive input vectors $\{\boldsymbol{x}_t\}_{t\geq 1}$, $\boldsymbol{x}_t \in \mathbb{R}^d$ and $\{d_t\}_{t\geq 1}$, $d_t \in \mathbb{R}$ sequentially. Our aim is to estimate $\hat{d}_t = \boldsymbol{p}_t^T \boldsymbol{x}_t$, where $\boldsymbol{p}_t \in \mathbb{R}^d$. We then calculate the loss $\ell_t(d_t, \hat{d}_t)$ and update the corresponding parameters of the regression model. In this paper, to estimate $\hat{d}_t$, besides using present sample $\boldsymbol{x}_t$, we use both present and past samples, i.e., $\{\boldsymbol{x}_{t-j}\}$ for $j = 1, \ldots, \nu$, where ν corresponds to the

index of the past sample till which we want to use. We use recurrent neural networks (RNNs) to execute such a task. The basic framework of a simple RNN is given as follows [9]:

$$\boldsymbol{h}_t = \kappa(\boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{R}\boldsymbol{h}_{t-1}) \tag{1}$$

$$\boldsymbol{y}_t = \psi(\boldsymbol{h}_t), \tag{2}$$

where $\boldsymbol{x}_t \in \mathbb{R}^d$ is the input vector, $\boldsymbol{h}_t \in \mathbb{R}^m$ is the RNN state vector and $\boldsymbol{y}_t \in \mathbb{R}^m$ is the output vector. The functions $\kappa(\cdot)$ and $\psi(\cdot)$ are commonly set to $\tanh(\cdot)$ and applies pointwise to vectors. The coefficient weight matrices $\boldsymbol{W}$ and $\boldsymbol{R}$ are set to appropriate dimensions.

In this paper, we apply FFT-IFFT operation on the RNN framework. Let $\boldsymbol{F}_N^{(nk)}$ and $\boldsymbol{F}_N^{*(nk)}$ denote the Fourier and inverse-Fourier matrices respectively where $0 \leq n, k \leq d-1$ and N is the length of the data sequence vector or row size of the matrix to which it is applied. Let us define an FFT operation function $\theta_F(\boldsymbol{a}, \boldsymbol{b})$ as

$$\theta_F(\boldsymbol{a}, \boldsymbol{b}) := (\boldsymbol{F}_N^{(nk)}\boldsymbol{a})(\boldsymbol{F}_N^{(nk)}\boldsymbol{b}), \tag{3}$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ can be a vector or a matrix. We apply the FFT-IFFT operation on (1) as follows:

$$\boldsymbol{h}_t = \kappa\Big(\boldsymbol{F}_m^{*(nk)}\big(\theta_F(\boldsymbol{W}, \boldsymbol{x}_t) + \theta_F(\boldsymbol{R}, \boldsymbol{h}_{t-1})\big)\Big). \tag{4}$$

As a special case of the RNNs, we use the GRU neural networks [11] with one hidden layer with FFT-IFFT operation defined as follows:

$$\begin{aligned}
\boldsymbol{z}_t &= \sigma\Big(\boldsymbol{F}_m^{*(nk)}\big(\theta_F(\boldsymbol{W}^{(z)}, \boldsymbol{x}_t) + \theta_F(\boldsymbol{U}^{(z)}, \boldsymbol{h}_{t-1})\big)\Big) \\
\boldsymbol{r}_t &= \sigma\Big(\boldsymbol{F}_m^{*(nk)}\big(\theta_F(\boldsymbol{W}^{(r)}, \boldsymbol{x}_t) + \theta_F(\boldsymbol{U}^{(r)}, \boldsymbol{h}_{t-1})\big)\Big) \\
\tilde{\boldsymbol{h}}_t &= g\Big(\boldsymbol{F}_m^{*(nk)}\big(\theta_F(\boldsymbol{W}^{(h)}, \boldsymbol{x}_t) + \theta_F(\boldsymbol{U}^{(h)}, \\
&\qquad \theta_F(\Delta^{(\boldsymbol{r}_t)}, \boldsymbol{h}_{t-1}))\big)\Big) \\
\boldsymbol{h}_t &= \boldsymbol{F}_m^{*(nk)}\big(\theta_F(\Delta^{(\mathbf{1}-\boldsymbol{z}_t)}, \tilde{\boldsymbol{h}}_t) + \theta_F(\Delta^{(\boldsymbol{z}_t)}, \boldsymbol{h}_{t-1})\big),
\end{aligned} \tag{5}$$

where $\boldsymbol{x}_t \in \mathbb{R}^d$ is the input vector and $\boldsymbol{h}_t \in \mathbb{R}^m$ is the output vector. $\boldsymbol{z}_t$ and $\boldsymbol{r}_t$ are the reset and update gates respectively. $\Delta^{(\cdot)}$ is a diagonal matrix and the coefficient weight matrices, i.e., $\boldsymbol{W}^{(\cdot)}$ and $\boldsymbol{U}^{(\cdot)}$, are set to appropriate dimensions. The functions $\sigma(\cdot)$ and $g(\cdot)$ are set to sigmoid and $\tanh$ respectively and apply pointwise to vectors. The FFT-IFFT operation on the GRU network is shown in Fig. 1.

A. Gradient Based Additive Updates

In this subsection, we derive the gradient based additive updates of the GRU network parameters with one hidden layer with FFT-IFFT operation. For the loss function $\ell_t(d_t, \hat{d}_t) = (d_t - \hat{d}_t)^2$, the stochastic gradient descent (SGD) update for parameter τ_{pq} (p^{th} row and q^{th} column entry of $\boldsymbol{\tau}$) is given as follows:

$$\tau_{pq} = \tau_{pq} - \eta\frac{\partial \ell_t}{\partial \tau_{pq}} = \tau_{pq} + 2(d_t - \hat{d}_t)\boldsymbol{p}_t^T\frac{\partial \boldsymbol{h}_t}{\partial \tau_{pq}} \tag{6}$$

where η is the learning rate.

For the GRU network defined in (5) and for $\tau_{pq} = w_{pq}^{(z)}$, $\dfrac{\partial \boldsymbol{h}_t}{\partial \tau_{pq}}$ is calculated as follows:

$$\begin{aligned}
\frac{\partial \boldsymbol{h}_t}{\partial w_{pq}^{(z)}} &= \boldsymbol{F}_m^{*(nk)}\Big(\theta_F\big(\Delta^{(\boldsymbol{\alpha})}, \tilde{\boldsymbol{h}}_t\big) + \theta_F\big(\Delta^{(\mathbf{1}-\boldsymbol{z}_t)}, \Delta^{(\boldsymbol{\gamma})}\big) + \\
&\qquad \theta_F\big(\Delta^{(\boldsymbol{\psi})}, \boldsymbol{h}_{t-1}\big) + \theta_F\big(\Delta^{(\boldsymbol{z}_t)}, \boldsymbol{\beta}_{t-1}^{(w_{pz}^{(z)})}\big)\Big)
\end{aligned} \tag{7}$$

$$\begin{aligned}
\frac{\partial \boldsymbol{z}_t}{\partial w_{pq}^{(z)}} &= \Delta^{\left(\sigma'(\cdot)\right)}\Bigg(\boldsymbol{F}_m^{*(nk)}\Big(\theta_F\big(\Delta^{(\boldsymbol{\phi})}, \boldsymbol{x}_t\big) + \theta_F\big(\boldsymbol{U}^z, \\
&\qquad \boldsymbol{\beta}_{t-1}^{(w_{pz}^{(z)})}\big)\Big)\Bigg)
\end{aligned} \tag{8}$$

$$\begin{aligned}
\frac{\partial \tilde{\boldsymbol{h}}_t}{\partial w_{pq}^{(z)}} &= \Delta^{\left(\tanh'(\cdot)\right)}\Bigg(\boldsymbol{F}_m^{*(nk)}\Big(\theta_F\big(\boldsymbol{U}^{(h)}, \theta_F\big(\Delta^{(\boldsymbol{\zeta})}, \boldsymbol{h}_{t-1}\big)\big) \\
&\qquad + \theta_F\big(\boldsymbol{U}^{(h)}, \theta_F\big(\Delta^{(\boldsymbol{r}_t)}, \boldsymbol{\beta}_{t-1}^{(w_{pz}^{(z)})}\big)\big)\Big)\Bigg)
\end{aligned} \tag{9}$$

$$\frac{\partial \boldsymbol{r}_t}{\partial w_{pq}^{(z)}} = \Delta^{\left(\sigma'(\cdot)\right)}\Bigg(\boldsymbol{F}_m^{*(nk)}\Big(\theta_F\big(\boldsymbol{U}_{(r)}, \boldsymbol{\beta}_{t-1}^{(w_{pz}^{(z)})}\big)\Big)\Bigg), \tag{10}$$

where $\boldsymbol{\alpha} = \Big(\dfrac{\partial(\mathbf{1}-\boldsymbol{z}_t)}{\partial w_{pq}^{(z)}}\Big)$, $\boldsymbol{\gamma} = \Big(\dfrac{\partial \tilde{\boldsymbol{h}}_t}{\partial w_{pq}^{(z)}}\Big)$, $\boldsymbol{\psi} = \Big(\dfrac{\partial \boldsymbol{z}_t}{\partial w_{pq}^{(z)}}\Big)$, $\boldsymbol{\phi} = \Big(\dfrac{\partial \boldsymbol{W}^{(z)}}{\partial w_{pq}^{(z)}}\Big) = \delta_{pq}^{(W^{(z)})}\boldsymbol{I}$, $\boldsymbol{\zeta} = \Big(\dfrac{\partial \boldsymbol{r}_t}{\partial w_{pq}^{(z)}}\Big)$ $\Delta^{(\mathbf{1}-\boldsymbol{z}_t)} = -\Delta^{(\boldsymbol{z}_t)}$ and $\boldsymbol{\beta}_{t-1}^{(w_{pz}^{(z)})} = \Big(\dfrac{\partial \boldsymbol{h}_{t-1}}{\partial w_{pq}^{(z)}}\Big)$. By substituting the value of (10) in (9), then (8) and (9) in (7) and finally substituting the value of (7) in (6), we can evaluate the online additive update at time t for the corresponding parameter. For the rest of the network parameters, the updates can be derived accordingly.

B. Weight Matrix Factorization (WMF) on FFT-IFFT based GRU Networks

In this subsection, we use the WMF on FFT-IFFT based GRU networks. We use the idea form [15] and apply to our model. Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ be a matrix, then by using WMF we can write $\boldsymbol{A} \approx \boldsymbol{B}\boldsymbol{C}$, where $\boldsymbol{B} \in \mathbb{R}^{m\times p}$, $\boldsymbol{C} \in \mathbb{R}^{p\times n}$ and $p << \min(m, n)$. We apply WMF on our FFT-IFFT based GRU network with $\boldsymbol{W}^{(\cdot)} \approx \boldsymbol{P}^{(\cdot)}\boldsymbol{Q}^{(\cdot)}$ and $\boldsymbol{U}^{(\cdot)} \approx \boldsymbol{R}^{(\cdot)}\boldsymbol{S}^{(\cdot)}$ as follows:

$$\begin{aligned}
\boldsymbol{z}_t = \sigma\Bigg(&\boldsymbol{F}_m^{*(nk)}\Big(\theta_F\big(\boldsymbol{P}^{(z)}, \theta_F(\boldsymbol{Q}^{(z)}, \boldsymbol{x}_t)\big) + \\
&\theta_F\big(\boldsymbol{R}^{(z)}, \theta_F(\boldsymbol{S}^{(z)}, \boldsymbol{h}_{t-1})\big)\Big)\Bigg)
\end{aligned} \tag{11}$$

$$\begin{aligned}
\boldsymbol{r}_t = \sigma\Bigg(&\boldsymbol{F}_m^{*(nk)}\Big(\theta_F\big(\boldsymbol{P}^{(r)}, \theta_F(\boldsymbol{Q}^{(r)}, \boldsymbol{x}_t)\big) + \\
&\theta_F\big(\boldsymbol{R}^{(r)}, \theta_F(\boldsymbol{S}^{(r)}, \boldsymbol{h}_{t-1})\big)\Big)\Bigg)
\end{aligned} \tag{12}$$

$$\begin{aligned}
\tilde{\boldsymbol{h}}_t = g\Bigg(&\boldsymbol{F}_m^{*(nk)}\Big(\theta_F\big(\boldsymbol{P}^{(h)}, \theta_F(\boldsymbol{Q}_{(h)}, \boldsymbol{x}_t)\big) + \\
&\theta_F\big(\boldsymbol{R}^{(h)}, \theta_F(\boldsymbol{S}^{(h)}, \theta_F(\Delta^{(\boldsymbol{r}_t)}, \boldsymbol{h}_{t-1})\big)\Big)\Bigg)
\end{aligned} \tag{13}$$

$$\boldsymbol{h}_t = \boldsymbol{F}_m^{*(nk)}\big(\theta_F(\Delta^{(\mathbf{1}-\boldsymbol{z}_t)}, \tilde{\boldsymbol{h}}_t) + \theta_F(\Delta^{(\boldsymbol{z}_t)}, \boldsymbol{h}_{t-1})\big), \tag{14}$$

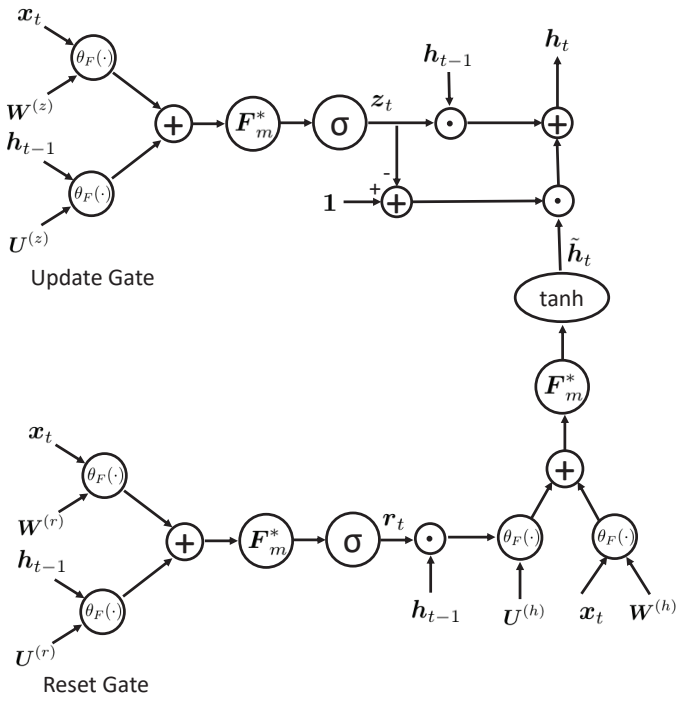


Fig. 1: Detailed schematic diagram of the FFT-GRU network.

Remark 1: *For the case of WMF based FFT-IFFT GRU networks, the gradient updates are similar to the updates for FFT-IFFT GRU networks as shown in the previous section in (7)-(10). For the GRU network with WMF defined in (11)-(14) and for* $\tau_{pq} = p_{pq}^{(z)}$, $\dfrac{\partial \boldsymbol{h}_t}{\partial \tau_{pq}}$ *is calculated as follows:*

$$\frac{\partial \boldsymbol{h}_t}{\partial p_{pq}^{(z)}} = \boldsymbol{F}_m^{*(nk)}\Big(\theta_F\big(\Delta^{(\boldsymbol{\alpha})}, \tilde{\boldsymbol{h}}_t\big) + \theta_F\big(\Delta^{(\mathbf{1}-\boldsymbol{z}_t)}, \Delta^{(\boldsymbol{\gamma})}\big) + \theta_F\big(\Delta^{(\boldsymbol{\psi})}, \boldsymbol{h}_{t-1}\big) + \theta_F\big(\Delta^{(\boldsymbol{z}_t)}, \boldsymbol{\beta}_{t-1}^{(p_{pz}^{(z)})}\big)\Big) \tag{15}$$

$$\frac{\partial \boldsymbol{z}_t}{\partial p_{pq}^{(z)}} = \Delta^{\left(\sigma'(\cdot)\right)}\Bigg(\boldsymbol{F}_m^{*(nk)}\Big(\theta_F\big(\Delta^{(\boldsymbol{\phi})}, \theta_F\big(\boldsymbol{Q}^{(z)}, \boldsymbol{x}_t\big)\big) + \theta_F\big(\boldsymbol{R}^z, \theta_F\big(\boldsymbol{S}^z, \boldsymbol{\beta}_{t-1}^{(p_{pz}^{(z)})}\big)\big)\Big)\Bigg) \tag{16}$$

$$\frac{\partial \tilde{\boldsymbol{h}}_t}{\partial p_{pq}^{(z)}} = \Delta^{\left(g'(\cdot)\right)}\Bigg(\boldsymbol{F}_m^{*(nk)}\Big(\theta_F\big(\boldsymbol{R}^{(h)}, \theta_F\big(\boldsymbol{S}^{(h)}, \theta_F\big(\Delta^{(\boldsymbol{\zeta})}, \boldsymbol{h}_{t-1}\big)\big)\big) + \theta_F\big(\boldsymbol{R}^{(h)}, \theta_F\big(\boldsymbol{S}^{(h)}, \theta_F\big(\Delta^{(\boldsymbol{r}_t)}, \boldsymbol{\beta}_{t-1}^{(p_{pz}^{(z)})}\big)\big)\big)\Big)\Bigg) \tag{17}$$

$$\frac{\partial \boldsymbol{r}_t}{\partial p_{pq}^{(z)}} = \Delta^{\left(\sigma'(\cdot)\right)}\Bigg(\boldsymbol{F}_m^{*(nk)}\Big(\theta_F\big(\boldsymbol{R}_{(r)}, \theta_F\big(\boldsymbol{S}_{(r)}, \boldsymbol{\beta}_{t-1}^{(p_{pz}^{(z)})}\big)\big)\Big)\Bigg), \tag{18}$$

where $g^{'}(\cdot) = \tanh^{'}(\cdot)$. Similarly, substituting the value of (18) in (17), (16) and (17) in (15) and then finally substituting the value of (15) in (6) we can calculate the online additive update of the corresponding parameter. For rest of the network parameters, we can similarly derive the updates as in (15)-(18).

III. Experiments

In this section, we illustrate the performance of our proposed algorithm on various data sets. We first compare the regression performance of our algorithm on the financial data set, i.e., Alcoa corporation stock price data set [16]. We then evaluate the regression performance on various real-life data sets, i.e., kinematics [17] and elevators[18]. Throughout this section, we use four variants namely: simple GRU, FFT-GRU, WMF-GRU and FFT-WMF-GRU networks. We also validate the performance by using the LSTM networks.

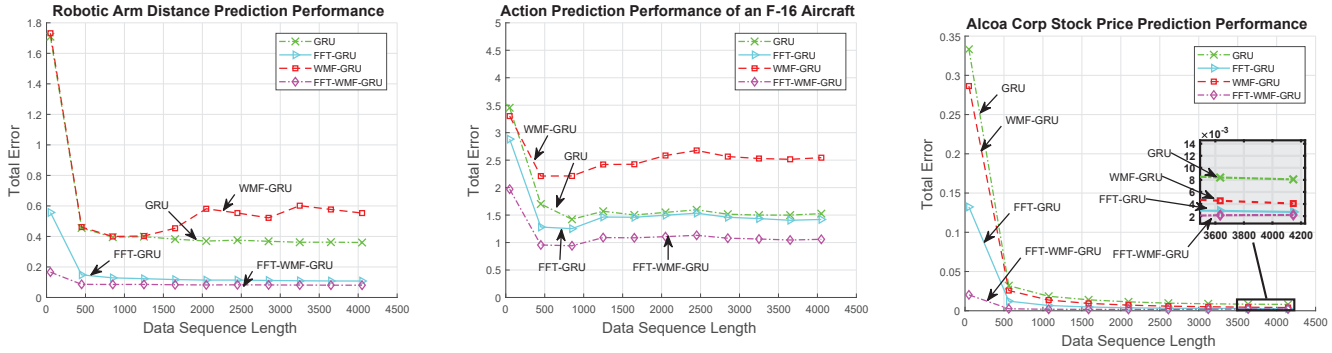
In Fig. 2, we compare the regression performance by calculating the time accumulated error over all the data sets. In our experimental setup, we use learning rate for SGD updates to be $\eta = 0.1$ and keep the common parameters of all the variants same for a fair experimental environment. For the WMF, we use $rank = 2$ for all the network weight sub-matrices. Fig. 2 shows the regression performance using simple, FFT and WMF variants of the GRU networks. Fig. 2a illustrates the distance prediction performance for the 8-link all revolute robotic arm. The input for this data set is $\boldsymbol{x} \in \mathbb{R}^8$. The main aim of this robot architecture is to predict the distance of the disturbing medium, i.e., effector, form the target. In Fig. 2b, we predict the set of actions taken by the F-16 aircraft with respect to various parameters related to the aircraft mechanism. Here, the input vector is $\boldsymbol{x} \in \mathbb{R}^{18}$. Similarly, in Fig. 2c, we have the daily stock price data for the Alcoa Corporation. We aim to predict the next instant future stock price by relying on the past values of the stock price data. As a whole, in Fig. 2, we observe that FFT-WMF-LSTM shows the best performance among all other variants like FFT-GRU, WMF-GRU and GRU networks among all the data sets. We also perform the same set of experiments using the LSTM networks.

Table. I, shows the steady state errors for both of the GRU and LSTM variants for all three data sets. From Table. I, we observe that the FFT-WMF-GRU networks surpass all other WMF and FFT GRU variants as well as the LSTM network variants. In Table. II, we list down the total number of the network parameters to be trained for all the variants of the algorithms and for all three data sets. From Table. I and Table. II, we can conclude that the FFT-WMF-GRU/LSTM networks have the lowest steady state error as well as less number of network training parameters.

Remark 2: *The effect of WMF on the network can be more precisely seen for the Elevators data set, where the number of parameters for the simple and WMF variant GRU network is 2034 and 306 respectively. There is* $\approx 6.5\%$ *reduction in the number of network parameters but we still manage to get the best performance as shown in Table. I.*

IV. Conclusions

We analyzed the online regression performance on the GRU networks using FFT-IFFT operation. We performed the gating operations in the frequency domain and then converted them back to time domain before the non-linear functions in the GRU networks. We first derived the online additive updates for FFT-GRU networks and then validated their performance



(a) Distance prediction performance of a robotic arm.

(b) Action prediction performance of an F-16 Aircraft.

(c) Alcoa Corporation stock price performance.

Fig. 2: Regression performance of Kinematics, Elevators and Alcoa Corp. data sets using simple and FFT, WMF variants of the GRU network.

TABLE I: Steady State Error Performance of simple and FFT, WMF variants of the GRU and LSTM networks.

Data Sets/Algorithms	GRU	FFT-GRU	WMF-GRU	FFT-WMF-GRU	LSTM	FFT-LSTM	WMF-LSTM	FFT-WMF-LSTM
Kinematics	0.3602	0.1077	0.5533	**0.0804**	0.3599	0.9991	0.3826	0.0815
Elevators	1.5260	1.4200	2.5450	**1.0590**	1.5255	1.3905	2.5555	1.0637
Alcoa Corp.	0.0080	0.0026	0.0040	0.0021	0.0079	0.0026	0.0043	**0.0020**

TABLE II: Total number of network parameters for the simple and FFT, WMF variants of the GRU and LSTM networks

Data Sets/ Algorithms	Kinematics	Elevators	Alcoa
GRU / FFT-GRU	424	2034	155
WMF-GRU / FFT-WMF-GRU	136	306	65
LSTM / FFT-LSTM	560	2682	225
WMF-LSTM / FFT-WMF-LSTM	168	378	105

on both financial and real-life data sets. We then applied the WMF on the FFT-GRU networks and then derived the online additive updates for one of the parameters. Furthermore, we demonstrate significant performance improvement in terms of cumulative error for the FFT-WMF-GRU networks. We also perform similar experiments using the LSTM architecture and then compare their results. We observed that our proposed FFT-WMF based GRU and LSTM networks performs much better than the simple GRU/LSTM, WMF-GRU/LSTM and FFT-GRU/LSTM networks and are much less computationally expensive in terms of number of network parameters. We observed that the GRU based networks are superior to LSTM based networks with FFT-IFFT operation.

REFERENCES

[1] A. C. Singer, G. W. Wornell, and A. V. Oppenheim, "Nonlinear autoregressive modeling and estimation in the presence of noise," *Digital signal processing*, vol. 4, no. 4, pp. 207–221, 1994.

[2] I. Ali and Y.-T. Chen, "Design quality and robustness with neural networks," *IEEE Transactions on Neural Networks*, vol. 10, no. 6, pp. 1518–1527, 1999.

[3] V. G. Vovk, "Aggregating strategies," *Proc. of Computational Learning Theory, 1990*, 1990.

[4] A. C. Tsoi, "Gradient based learning methods," in *Adaptive processing of sequences and data structures*. Springer, 1998, pp. 27–62.

[5] A. Krzyzak and T. Linder, "Radial basis function networks and complexity regularization in function learning," in *Advances in neural information processing systems*, 1997, pp. 197–203.

[6] S. Bengio and Y. Bengio, "Taking on the curse of dimensionality in joint distributions using neural networks," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 550–557, 2000.

[7] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[8] M. Hermans and B. Schrauwen, "Training and analysing deep recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2013, pp. 190–198.

[9] J. Mazumdar and R. G. Harley, "Recurrent neural networks trained with backpropagation through time algorithm to estimate nonlinear load harmonic currents," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 9, pp. 3484–3491, 2008.

[10] H. Jaeger, *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*. GMD-Forschungszentrum Informationstechnik, 2002, vol. 5.

[11] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[12] C. Van Loan, *Computational frameworks for the fast Fourier transform*. Siam, 1992, vol. 10.

[13] E. O. Brigham and E. Brigham, *The fast Fourier transform and its applications*. prentice Hall Englewood Cliffs, NJ, 1988, vol. 1.

[14] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons, "Algorithms and applications for approximate nonnegative matrix factorization," *Computational statistics & data analysis*, vol. 52, no. 1, pp. 155–173, 2007.

[15] O. Kuchaiev and B. Ginsburg, "Factorization tricks for lstm networks," *arXiv preprint arXiv:1703.10722*, 2017.

[16] Alcoa Inc. *Common Stock*. Accessed: Oct. 1, 2016. [Online]. Available: http:// finance.yahoo.com/quote/AA?ltr=1

[17] C. E. Rasmussen *et al.*, *Delve data sets*. Accessed: Oct. 1, 2016. [Online]. Available: http://www.cs.toronto.edu/delve/data/datasets.html

[18] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework." *Journal of Multiple-Valued Logic & Soft Computing*, vol. 17, 2011.