Adapting Random Simple Recurrent Network for Online Forecasting Problems

Mohammed Elmahdi Khennour [*], Abdelhamid Bouchachia[†], Mohammed Lamine Kherfi[*‡], Khadra Bouanane[*] and Oussama Aiadi[*]
[*] Laboratory of Artificial Intelligence and Information Technology, Kasdi Merbah University, Ouargla, Algeria.
[†] Dept of Computing & Informatics, Bournemouth University, Poole, UK.
[‡] LAMIA Laboratory, Université du Québec à Trois-Rivières, Canada.

Abstract—**Random Simple Recurrent Network (RSRN) is a forecasting model based on the Random Neural Network (RaNN) and Recurrent Neural Network (RNN). RSRN has demonstrated energy-efficient and effective forecasting capabilities in offline mode, making it suitable for various applications. However, offline training faces challenges, such as limited storage capacity, computational power, and evolving datasets. To address these limitations, this paper introduces an online learning approach to the RSRN model. We present adaptations of two online learning algorithms, Projected Online Gradient Descent (POGD) and Follow-The-Proximally-Regularized-Leader (FTRL-Proximal), for training RSRN in real-time. POGD leverages Back Propagation Through Time (BPTT) for handling dependencies with a sliding window, while FTRL-Proximal offers a balance between adaptability and stability, especially for sparse data. Our approach is the first to introduce RSRN's forecasting capabilities in a dynamic environment, demonstrating its potential in real-world applications where data availability is not guaranteed. The effectiveness of the online RSRN with both approaches is demonstrated through experimental results on benchmark datasets, showcasing competitive performance that surpasses offline mode computation and result.**

Keywords: Random Simple Recurrent Network, Online Learning, Forecasting problems, Projected Online Gradient Descent, Follow-The-Proximally-Regularized-Leader.

I. Introduction

Random Neural Network (RaNN) [1] is a mathematical model inspired by the biological behaviour of neurons in the brain cortex. It mimics the exchange of excitatory and inhibitory (positive and negative) signals between neurons. Since its inception by Gelenbe [1], RaNNs have been widely used for various applications such as network traffic monitoring [2], DDoS attack detection [3, 4, 5, 6], epileptic seizure classification [7], IoT network security assessment [8], class-incremental learning [9]. Several extensions of RaNN have been proposed such as multiple signals RaNN [10], recurrent RaNN [11] and synchronized interactions RaNN [12]. Recently, Yin [13] proposed several deep learning models based on RaNN such as CNN-based RaNN [14], auto-encoder-based RaNN [15], and Generative Adversarial Network-based (GAN) RaNN [16]. One of the recent variants of RaNN is the Random Simple Recurrent Network. RSRN [17] has been proposed as a forecasting model based on RaNN and recurrent NN.

RSRN achieved a comparable performance against state-of-art recurrent models regarding forecasting multi-variate sequential data with less training time and computational cost. This model proved to be energy-efficient making it suitable for power-limited environments or restricted data usage. RSRN was tested with various real-world applications such as temperature forecasting, oil price prediction, and energy consumption projection.

RSRN was developed in offline mode, which required the whole dataset to be available at the beginning of training. However, sometimes during offline training phase this is not possible due to various constraints, including insufficient storage capacity for data preservation, restricted computational power for training extensive datasets, limitations in energy resources or their costly usage, and potentially evolving datasets. Furthermore, time-series data often exhibits changing patterns and trends over time, non-stationary behaviour and long range dependencies. A viable resolution to the problems at hand is online learning. It is particularly advantageous for recurrent models including RSRN, which are designed to capture temporal dependencies and patterns in sequential data. Recurrent models, by nature, are well-suited for sequential data. Online learning allows these models to continuously adapt to new information as it arrives in real time. Also, It enables them to efficiently handle streaming data, where observations are received sequentially. Furthermore, Recurrent models, especially those dealing with long sequences, can benefit from memory-efficient online learning algorithms. These algorithms allow the model to update its internal state incrementally, avoiding the need to store and process the entire sequence at once.

Online Learning refers to the paradigm where a model is updated iteratively and adaptively over time as new data becomes available in a sequential manner. Unlike traditional batch learning, where the model is trained on the entire dataset at once, online learning processes data sequentially, updating the model parameters after each new observation. This continuous learning approach is particularly well-suited for dynamic and evolving datasets, enabling the model to adjust its predictions in real-time and efficiently handle streaming or changing data streams. Online learning algorithms aim to strike a balance between exploiting newly acquired information and retaining knowledge from past observations, making them suitable for applications in scenarios with non-stationary data distributions and the need for real-time responsiveness.

Several approaches are commonly used in an online learning setting. Projected Online Gradient Descent POGD is a famous approach generalized from the standard GD to accommodate online requirements. Forward-Backward Splitting [18] is also another algorithm for online learning. This approach adjusts the model parameters and applies a sparsity-inducing penalty to encourage some model parameters to be exactly zero. The Regularized Dual Averaging [19] method adapts the model parameters gradually while incorporating a regularization term to induce sparsity in the model. Follow-The-Leader [20, 21] and its extension (Follow-The-Regularized-Leader FTRL) [22] is widely used for online approaches with sparse and non-stationary data. FTRL combines the idea of following the leader (learning from a leading example) in adjusting model parameters with regularization techniques to prevent overfitting. Follow-The-Proximally-Regularized-Leader (FTRL-Proximal) is yet a recent extension of the FTRL based on proximal term that controls the learning rate. Online learning techniques are tailored to handle the evolving, sequential, and real-time nature of time-series data, providing a flexible and adaptive approach to process dynamic temporal patterns.

Our approach introduces the online setting to RSRN forecasting model. We adapted POGD and FTRL-Proximal to train RSRN adaptively. POGD will adapt the Back Propagation approach used by RSRN for training. However, it will not hold any dependencies. For this reason, we can use Back Propagation Through Time (BPTT) to handle those dependencies. This approach will require a data history, thus, we will add a sliding window that holds some temporal data. Another strategy called FTRL-Proximal designed for sparse data. Its process consist of accumulating the gradient of previous observation, which resemble the concept of BPTT. This approach stores past dependencies with less memory than BPTT, this make it appropriate for recurrent model such as RSRN. FTRL-Proximal uses per-coordinate learning rate as proximal term to offer a balance between adaptability and stability during training.

A. *Related Works*

Online learning for forecasting applications has a significant importance regarding predicting evolving patterns and behaviours in sequential data. Generally, Most of these works build upon recurrent models to capture temporal dependencies with various adaptation and techniques. Fekri et al. [23] uses an online adaptive RNN (LSTM) model for load forecasting in addition to preprocessing, buffering, and tuning modules. Guo et al. [24] uses the weighted gradient learning approach on RNN model by leveraging local features to adapt to changes and resist outliers. Yang et al. [25] adapts Robust Adam to update LSTMs parameters and adaptively tunes the learning rate. Zucchet et al. [26] adapts Linear Recurrent Units as a learner to reduce complexity of RNNs. Despite using RNNs and mostly LSTM for online forecasting applications which yield formidable performance, they are computationally expensive which will impact their energy footprint.

Random NN is used for various applications for more than two decades for its energy-efficient architecture [27]. However, its utilization in the online setting is very limited and only focuses on cybersecurity and intrusion detection as far as we know. Nakip et al. [28] uses auto encoder RaNN as a base model for decentralized and online federated learning intrusion detection. The work of [29] is based on online self-supervised RaNN for intrusion detection in IoT environment. Also, [30] utilizes deep RaNN model for real-time cyberattack detection with online and offline setting. Mostly, all the adaptive RaNN models works for classification problems. Meanwhile, as far as we know, there is not any works regarding online RaNN for forecasting problems. Our work is the first to introduce the online setting to the RaNN (and by extension RSRN) regarding forecasting problems.

The next section II will provide a background description of the RSRN model architecture and properties, Then it will give the adapted online approaches and its algorithms in detail. In section III, we validate our approach with different algorithms and setting. Lastly in section IV, we conclude our results and discuss further directions.

II. Online RSRN

A. *Random Simple Recurrent Network (RSRN): overview*

Random Simple Recurrent Network (RSRN) is a recurrent model based on mixture of Random Neural Network's (RaNN)[1] behaviour and neural architecture with Recurrent Neural Network's (RNN) [31, 32, 33] recurrency. RSRN proved its effectiveness regarding forecasting sequential multivariate data even with long range dependencies. This model achieved a comparable performance against state-of-art recurrent models with low computational cost and less training time. Figure 1 represents a 3-layer RSRN network, where the hidden layer in the model has a context layer that saves the temporal dependencies of previous data and passes it at the next iteration. RSRN is an extension of the RaNN model, hence it has the same properties and characteristics with some modification to accommodate the recurrent topology of RNN. RSRN basically contains Random Neurons (RN) which is a type of neurons that receives positive and negative signals from previous neurons and the environment. The positive/negative signals could be expressed as positive weights ($W^+/W^- \geq 0$). A RN i from a layer I may receive signals from previous neurons j in layer J as ϱ_j or from the environment Λ (positive signals) and λ (negative signals). The output of neuron i is called the stationary probability distribution (could be considered as an activation function) ϱ_i expressed as follows

$$\varrho_i = \frac{T_i^+}{r_i + T_i^-} = \frac{\Lambda_i + \sum_{j=1}^{J} \varrho_j w_{j,i}^+ + \sum_{c=1}^{C} \varrho_c w_{c,i}^+}{r_i + \lambda_i + \sum_{j=1}^{J} \varrho_j w_{j,i}^- + \sum_{c=1}^{C} \varrho_c w_{c,i}^-} \tag{1}$$

where T_i^+ and T_i^- are the total arrival of positive and negative signals from the environment and the previous neurons J and context layers C. r_i is the firing rate calculated as the total of

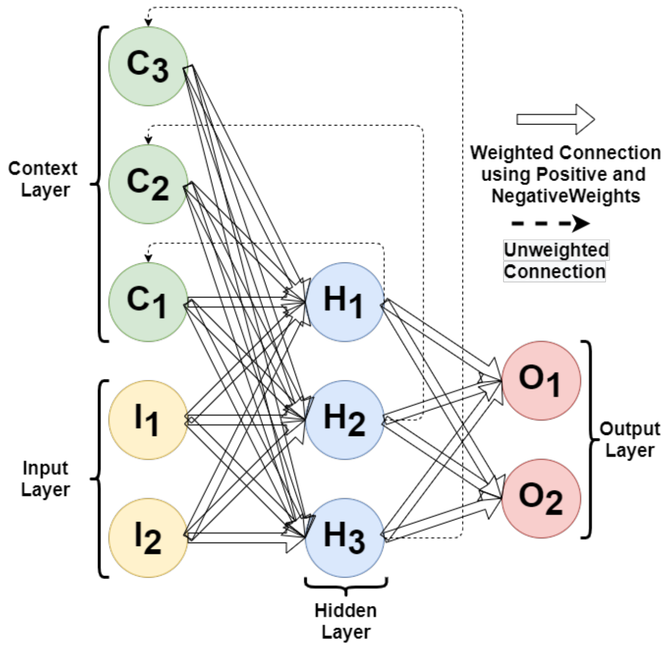

Fig. 1. RSRN architecture, where the connection between neurons simulate positive/negative signals and the context layer saves temporal dependencies[17]

the outgoing weights from neuron i to other neurons S, $r_i = \sum_s^S (W_{i,s}^{+} + W_{i,s}^{-})$. Λ_i and λ_i are the excitatory and inhibitory signal from the environment, also they are considered as the input data associated with neurons in the input layer. For each data input x_i:

$$\Lambda_i = \begin{cases} x_i & if x_i > 0 \\ 0 & otherwise \end{cases} \quad (2) \qquad \lambda_i = \begin{cases} |x_i| & if x_i < 0 \\ 0 & otherwise \end{cases} \quad (3)$$

$w_{j,i}^{+}$ and $w_{j,i}^{-}$ are the positive and negative connection weights between neuron j in layer J and i in layer I. $w_{c,i}^{+}$ and $w_{c,i}^{-}$ are the positive and negative connection weights between nodes c in the context layer C and other nodes i in layer I. If the neurons keep firing positive signals, the next neurons will saturate. To prevent such a saturation, a limit of firing is set as [34]:

$$\varrho_i \leftarrow min(\varrho_i, 1) \quad (4)$$

where $min(a, b)$ produces the smallest number between a and b.

In a forecasting setting, the model will try to map an input data x_t to its output x_{t+1}, hence, $f(x_t) = x_{t+1}$. At each round $t \in [1 \dots T]$, we have an input data (vector) x_t and the model will try to predict $\hat{y}$ which represent an approximation of $y = x_{t+1}$.

Throughout this paper we will use the MSE as a loss function and an evaluation metric. MSE is defined as follow:

$$E(\varrho_o, y) = \frac{1}{2O} \sum_{o}^{O} (\varrho_o - y_o)^2 \quad (5)$$

where ϱ_o is the output of the network and y is the desired value. O in the number of neurons in the output layer.

Khennour et al. [17] proposed three approaches to train RSRN; Back Propagation BP, Back Propagation Through Time BPTT and Truncated Back Propagation Through Time TBPTT. The problem with the proposed approaches is that they work in offline manner, which mean they require the availability of dataset at that beginning of training. Given that RSRN operates as a forecasting model, acquiring time-series data is not consistently feasible, and such data may be presented in a streaming manner. Consequently, the nature of this data necessitates an adaptable model capable of accommodating its dynamic conditions. Online learning approaches offer a fitting environment tailored to these particular requirements.

B. Online Learning for RSRN

RSRN is an energy-efficient recurrent model could benefit from online adaptability and non-stationarity. The adaptation will consist of redefining the process of updating the model parameters. Meanwhile its architecture and internal properties will remain the same. In the following two sections, we will provide two strategies to adapt RSRN as an online model.

C. Adapted Projected Online Gradient Descent with a sliding window K and training approach

Projected Online Gradient Descent (POGD) is an online optimization algorithm used to adapt to incoming data points one at a time and iteratively update model parameters based on the gradient of a loss function. The usual update rule is defined as $w_t = w_{t-1} - \eta \frac{\partial E}{\partial w_{t-1}}$, where η is the learning rate and $\frac{\partial E}{\partial w_{t-1}}$ is the gradient of the loss function E w.r.t the weight w. In addition, POGD includes a projection step that ensures the updated parameters remain within the feasible set, where a function $\Pi_{\mathbb{X}}(w)$ will project the parameter w into the feasible set $\mathbb{X}$ after each update. To adapt POGD to RSRN, we have to define the feasible set. RSRN's weight must be positive, thus our feasible set is $\mathbb{R}^{+}$. BP is the training approach suitable for this case, but for longer sequences, it may lose some dependencies. Hence, we propose a sliding window that saves $K-$last input data. This window will work as associative memory to help the model guard some temporal dependencies. The usage of this window will enable the RSRN model to use BPTT approach for training to enhance performance. At each step, the model will consider the gradient from current sample t to $t - K$. This will help the model to become insensitive to outliers and capable of capturing temporal dependencies.

For a multi layer RSRN network like in Fig. 1, we have a model with $M-$layer. The Adapted POGD with a sliding window K training algorithm is summarized in the Algo. 1 below. At each round t, we receive an input vector x_t and pass through the network. Then calculate the error and gradient and update the weights. lastly, we project our weight into our feasible set. $\Pi_{\mathbb{R}^{+}}(\cdot)$ is the projection of the weights in $\mathbb{R}^{+}$, we will define it as $\Pi_{\mathbb{R}^{+}}(W) = max(0, W)$. The parameter *Max_Trunc* K represent the size of temporal dependencies considered in the update as sliding window. In this approach,

we select a decreasing learning rate to offer a stability in the training process and to be resistant against outliers.
For the BPTT approach, we will use average of the gradient instead of the sum to prevent gradient explosion. Equation 6 gives the formula to calculate the average gradient over $K-$last gradient, where K is the size of the sliding window

Algorithm 1 Adapted POGD with sliding window K

```
Require: RSRN layers $m \in [1 \dots M]$, Input data $x_t, t \in [1 \dots T]$, Max_Trunc $K$
Ensure: Initialize Weights $W^+/W^- \geq 0$
  for $t = 1 \dots T$ do
      Receive input vector $x_t$
      Calculate firing rates $r_i$
      $\Lambda \leftarrow max(0, x_t)$
      $\lambda \leftarrow |min(0, x_t)|$
      $y \leftarrow x_{t+1}$
      Calculate $\varrho_{i,m}, i \in m, m \in [1 \dots M]$ using Eqs. 1
      Calculate Error $E_t(\varrho_M, y)$ using Eqs. 5
      Calculate Derivatives $\frac{\partial E_t}{\partial W^{+/-}_{t-K}}$ using the Average of BPTT using Eqs. 6
      update learning rate $\eta \leftarrow \frac{1}{\sqrt{t}}$
      $W^{+/-}_{t+1} \leftarrow \Pi_{\mathbb{R}^+}(W^{+/-}_t - \eta \frac{\partial E_t}{\partial W^{+/-}_{t-K}})$
  end for
```

$$\frac{\partial E_t}{\partial W_{t-K}} = \frac{1}{K} \sum_{s=t}^{t-K} \frac{\partial E_t}{\partial W_s} \tag{6}$$

According to Algo. 1, the size of the window K has an important factor regarding the performance of the model computationally and empirically. To study this effect, we consider 3 cases of K. When $K = 1$, this is the usual online mode, where at each round t, we receive one sample x_t and consider just that sample for model update. In this case, we can only use BP approach to update the model since we do not have temporal dependencies to consider.
When $K = constant > 1$, at each round t, we receive one sample x_t and consider the last $K-$samples to update the model. In this case, we could use BP or BPTT approaches to update it. This scenario could be expressed as sliding window K, where it preserves the temporal dependencies of size K allowing the model to access the history of data. We could further discuss the effect of different sizes of K in the experiment (see section III-B).
When $K = t$, where t is the number of samples presented until the current round. In this case, the model will have access to full history of data from the first sample x_1 to the current sample x_t. This scenario could be viewed as the offline mode with incremental manner, where the size of the data increases at each round. This scenario might not be considered as online learning but it is of paramount to study this case.

D. Adapted Follow The Proximally Regularized Leader

Follow The Proximally Regularized Leader (FTRL-Proximal) [22] is a variant of the FTRL algorithm family. It is a popular algorithm and widely used for the online setting including ad click prediction [35], product demands prediction [36] and online malware detection [37]. FTRL is similar to the POGD but with addition to L1 and L2 regularization terms and proximal term in the objective function. The usage of these regularizations induces sparsity, so it is beneficial in large-scale and sparse datasets. The proximal term in FTRL-Proximal helps to control the step size in the parameter updates by preventing the algorithm from making excessively large updates. It adds a form of stability to the learning process, particularly in scenarios where the data distribution may change over time or when dealing with sparse features. McMahan et al. [35] uses per-coordinate learning rate schedule as proximal term, where each coefficient has its own non-increasing learning rate. The objective function for the FTRL-Proximal algorithm is to minimize the following:

Algorithm 2 FTRL-Proximal with L1 and L2 regularization using per-coordinate learning rate for RSRN

```
Require: RSRN layers $m \in [1 \dots M]$, Parameters $\alpha, \beta, L_1, L_2$
Ensure: Initialize Weights $W_i^+/W_i^- \geq 0, z_i = 0, n_i = 0; \forall i \in m, m \in [1 \dots M]$
  for $t = 1 \dots T$ do
      for all $i \in m, m \in [1 \dots M]$ do
          $w^{+/-}_{t,i} = \begin{cases} 0 & if |z_i| \leq L_1 \\ -(\frac{\beta + \sqrt{n_i}}{\alpha} + L_2)^{-1}(z_i - sgn(z_i) L_1) & otherwise \end{cases}$
      end for
      Receive input vector $x_t$
      Calculate firing rates $r_i, \forall i \in m, m \in [1 \dots M]$
      $\Lambda_t \leftarrow max(0, x_t)$
      $\lambda_t \leftarrow |min(0, x_t)|$
      $y_t \leftarrow x_{t+1}$
      Calculate $\varrho_i, i \in m, m \in [1 \dots M]$ using Eqs. 1
      Calculate Error $E_t(\varrho_M, y)$ using Eqs. 5
      for all $i \in m, m \in [1 \dots M]$ do
          $g_i = \frac{\partial E_{t,i}}{\partial w^{+/-}_{t,i}}$ # gradient of the loss w.r.t $w^{+/-}_{t,i}$
          $\sigma_i = \frac{1}{\alpha}(\sqrt{n_i + g_i^2} - \sqrt{n_i})$
          $z_i \leftarrow z_i + g_i - \sigma_i w_{t,i}$ # Cumulative gradient
          $n_i \leftarrow n_i + g_i^2$
      end for
  end for
```

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} (\sum_{s=1}^{t} g_s \cdot \mathbf{w} + \frac{1}{2} \sum_{s=1}^{t} \sigma_s ||\mathbf{w} - \mathbf{w}_s||_2^2 + L_1 ||\mathbf{w}||_1) \tag{7}$$

where g_t is the gradient of the loss function at step t. L_1 is the strength of the L1-regularization. σ_t is a scaling term defined by the learning rate schedule $\sigma_t = \frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}$, hence $\sum_{s=1}^{t} \sigma_s = \frac{1}{\eta_t}$. Since we are using per-coordinate learning rate, $\eta_{t,i}$ is the learning rate of the ith parameter at step t

defined as follow:

$$\eta_{t,i} = \frac{\alpha}{\beta + \sqrt{\sum_{s=1}^{t} g_{s,i}^2}} \tag{8}$$

where α and β are hyper-parameters that controls the step size during parameters update.

A quick analysis for Eqs. 7 indicates that the gradient of previous steps has to be stored, but a reformulation of Eqs. 7 can help us optimize the process by storing just a variable per coefficient. The reformulation goes as follow:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \Big((\sum_{s=1}^{t} g_s - \sum_{s=1}^{t} \sigma_s \mathbf{w}_s) \cdot \mathbf{w} + \frac{1}{\eta_t} ||\mathbf{w}||_2^2 + L_1 ||\mathbf{w}||_1 + (const) \Big) \tag{9}$$

In this particular case, we can use z_i to store the accumulated gradient at each step. This is slightly similar to the BPTT process (Eqs.6), since both have to store previous gradients. However, in this case, the gradient is accumulated only once and not revisited at each update like BPTT do. The use of FTRL-Proximal in time-series forecasting with long range dependencies will yield formidable results.

Algorithm 2 describes the FTRL-Proximal process with per-coordinate learning rate schedule. This algorithm is inspired from McMahan et al. [35] work and adapted for the RSRN network with time-series forecasting setting. In this scenario, we will have an RSRN multi layer network with $M-$layer. The Algo. 2 will have α, β, L_1 and L_2 as hyper-parameters. At each step t, it will update its coefficients (weights) and receive an input vector x_t. After passing through the RSRN network, we calculate the error and the gradient of the current step. For each coefficient i, we will update its learning rate σ_i and accumulate the gradient into z_i and its square into n_i.

III. Experiment & Results

In this section, we will validate the model presented in the previous section and experiment with different scenarios of training. The code was implemented from scratch using python and was simulated in a laptop with i5 8th Gen, 16GB of RAM and 4GB NVIDIA GTX 1050. According to the empirical results, we selected a 3-layer RSRN network with 5 nodes in the hidden layer (input and output layer is equal to the feature size of the dataset) to validate throughout the following experiments. The MSE is used as loss function and we monitor it as the training proceed.

To evaluate our model and approaches, we used three forecasting datasets. Fuel dataset [38] has 10 features with 880 sample. It show the weekly fuel prices and their changes since 2013. Temperature dataset [39] has 8 features with 3180 sample. It measures the Global Land/Ocean Temperature from 1750 to 2015. GridWatch dataset [40] has 8 feature with nearly 796K sample. It describes the Britain's power grid sources. Throughout the following experiments, we use 75% of each dataset as training set in online manner and 25% of them as testing set.

A. Validating different learning scenarios of Adapted POGD

In this experiment, we used Adapted POGD to train the RSRN model using different training scenarios and approaches against each other and offline RSRN model. Table I represent a summary of the results. We used in this experiment 3 scenarios (K $\in$ {1,100,t}) and the offline mode. The training approaches we used are BP and BPTT to update the model. Notice that BPTT can not be used where K=1 since there is no temporal dependencies to back propagate across.

A general observation indicate that the best performance was obtained when using K=t and BP approach even better than BPTT and offline mode's best result. For the case of K=100, BP performed better than BPTT even computationally. Throughout all cases, the BP performed better than BPTT this is due to the reason that temporal dependencies was guarded through the incremental inference of the model not by learning approach (BPTT).

B. Validating the FTRL-Proximal approach

In this experiment, we used the FTRL-Proximal method described in algo. 2 to train RSRN model. FTRL-Proximal is usually used for large scale sparse dataset, we introduces two types of data. The normal datasets and the sparse one. The sparse datasets are the same with normal one but we induced sparsity with factor of 35% (randomly multiply 35% of data content with 0). The following experiment investigate the result of FTRL-Proximal approach with RSRN model and the effect of sparse data with accuracy. Table II shows the MSE result of the FTRL-Proximal approach for the RSRN model using normal and sparse datasets. As we can see from table II, for fuel and temperature dataset the sparse data has slightly better result than normal one. However, for the GridWatch dataset the normal data exceeds the sparse one regarding accuracy. The FTRL-Proximal seems that it performs great for relatively small scale sparse data. Meanwhile, for large-scale data, it tend to favour accuracy over sparsity.

The second row shows the result of POGD with normal and sparse data. Overall, the normal data's performance is better than sparse one. Comparing the result of POGD against FTRL-Proximal, we see that the POGD surpasses the FTRL-Proximal generally in a massive scale data (GridWatch). The POGD's performance regarding sparse data is less than normal one. This indicate than POGD does not perform well under sparse data. Generally, both algorithms tend to achieve optimal performance when dealing with large-scale data, POGD for normal data and FTRL-Proximal for sparse one.

IV. Conclusion

In this paper, we proposed an online adaptation for RSRN model in time-series forecasting problems. The algorithms are based on an adaptation of the POGD and FTRL-proximal with the recurrent topology of the model. These algorithms proved its effectiveness regarding forecasting problems. POGD was developed with a sliding window of size K that saves K previous temporal dependencies. The role of this window is to guard some history to prevent the model from straying too

TABLE I
MSE COMPARISON RESULT OF DIFFERENT TRAINING SCENARIOS AND APPROACHES

Dataset	Fuel		Temperature		GridWatch (10K)	
Training Scenario	BP	BPTT	BP	BPTT	BP	BPTT
K = 1	0.85268		0.28219		0.03727	
K = 100	0.02497	1.6801	**0.03366**	0.817405	0.00423	0.0756
K = t	**0.00695**	1.70452	0.03391	1.31379	**0.00154**	0.79788
Offline mode	0.087	0.036	0.114	0.059	0.103	0.029

TABLE II
FTRL-PROXIMAL & POGD PERFORMANCE WITH NORMAL AND SPARSE DATA

Method	Dataset	Fuel	Temperature	GridWatch
FTRL-Proximal	Normal	0.8542	0.5783	0.0003
	Sparse	0.549	0.4174	0.0066
POGD	Normal	0.0444	0.1111	4.01e-5
	Sparse	0.4305	0.3744	0.0055

far when dealing with outliers or non-stationary data. FTRL-Proximal algorithm was used with per-coordinate learning that that governs the learning step of each coefficient. The acquired result shows that POGD for RSRN achieves better result when we increase the size of window K. However, this comes with an increase in the computational cost. On the other side, FTRL-Proximal achieved an acceptable performance when dealing with small data. Meanwhile, its performance increases for sparse data or large scale datasets. Sometimes these algorithms reach a performance even better than offline approach. The RSRN model proved to be an energy-efficient model better than competitor models, which make it appropriate for environment with limited computational capability or energy constraint.
These algorithms reach their optimal performance when dealing with massive-scale data. FTRL-Proximal is preferred for sparse data while POGD is suitable for non-zero data. RSRN as before has the same downside where it performs poorly on classification problems.
In our next literature, we aim to apply RSRN for task-specific application such as proactive models and solve the time-series classification problems. Also, we want to investigate its capabilities regarding continual learning approaches.

ACKNOWLEDGEMENT

A. Bouchachia is funded by the EC through the project "ExtremeXP" under Grant Agreement No. 101093164.

REFERENCES

[1] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural computation*, vol. 1, no. 4, pp. 502–510, 1989.
[2] P. Fröhlich and E. Gelenbe, "Optimal fog services placement in sdn iot network using random neural networks and cognitive network map," in *International Conference on Artificial Intelligence and Soft Computing*. Springer, 2020, pp. 78–89.
[3] G. Öke and G. Loukas, "A denial of service detector based on maximum likelihood detection and the random neural network," *The Computer Journal*, vol. 50, no. 6, pp. 717–727, 2007.
[4] M. Nakıp and E. Gelenbe, "Online self-supervised learning in machine learning intrusion detection for the internet of things," *arXiv preprint arXiv:2306.13030*, 2023.
[5] S. Yatish, V. Vinod, S. S. Pande, V. L. Narayana, N. Nishant, and P. Sivagurunathan, "Deep learning for attack detection in industrial iot edge devices," in *2023 8th International Conference on Communication and Electronics Systems (ICCES)*. IEEE, 2023, pp. 539–544.
[6] E. Gelenbe and M. Nakip, "Real-time cyberattack detection with offline and online learning," in *2023 IEEE 29th International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 2023, pp. 1–6.
[7] S. Y. Shah, H. Larijani, R. M. Gibson, and D. Liarokapis, "Epileptic seizure classification based on random neural networks using discrete wavelet transform for electroencephalogram signal decomposition," *Applied Sciences*, vol. 14, no. 2, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/2/599
[8] E. Gelenbe and M. Nakip, "Iot network cybersecurity assessment with the associated random neural network," *IEEE Access*, vol. 11, pp. 85 501–85 512, 2023.
[9] M. Zając, T. Tuytelaars, and G. M. van de Ven, "Prediction error-based classification for class-incremental learning," *arXiv preprint arXiv:2305.18806*, 2023.
[10] E. Gelenbe and J.-M. Fourneau, "Random neural networks with multiple classes of signals," *Neural computation*, vol. 11, no. 4, pp. 953–963, 1999.
[11] E. Gelenbe, "Learning in the recurrent random neural network," *Neural computation*, vol. 5, no. 1, pp. 154–164, 1993.
[12] E. Gelenbe and S. Timotheou, "Random neural networks with synchronized interactions," *Neural Computation*, vol. 20, no. 9, pp. 2308–2324, 2008.
[13] Y. Yin, "Deep learning with the random neural network and its applications," *arXiv preprint arXiv:1810.08653*, 2018.
[14] Y. Yin and E. Gelenbe, "Single-cell based random neural network for deep learning," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 86–93.
[15] ——, "Non-negative autoencoder with simplified random neural network," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–6.
[16] W. Serrano, "The deep learning generative adversarial

random neural network in data marketplaces: The digital creative," *Neural Networks*, 2023.

[17] M. E. Khennour, A. Bouchachia, M. L. Kherfi, and K. Bouanane, "Randomising the simple recurrent network: a lightweight, energy-efficient rnn model with application to forecasting problems," *Neural Computing and Applications*, vol. 35, no. 27, pp. 19 707–19 718, 2023.

[18] Y. Singer and J. C. Duchi, "Efficient learning using forward-backward splitting," *Advances in Neural Information Processing Systems*, vol. 22, 2009.

[19] L. Xiao, "Dual averaging method for regularized stochastic learning and online optimization," *Advances in Neural Information Processing Systems*, vol. 22, 2009.

[20] J. Hannan, "Approximation to bayes risk in repeated play," *Contributions to the Theory of Games*, vol. 3, pp. 97–139, 1957.

[21] A. Kalai and S. Vempala, "Efficient algorithms for online decision problems," *Journal of Computer and System Sciences*, vol. 71, no. 3, pp. 291–307, 2005.

[22] B. McMahan, "Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 525–533.

[23] M. N. Fekri, H. Patel, K. Grolinger, and V. Sharma, "Deep learning for load forecasting with smart meter data: Online adaptive recurrent neural network," *Applied Energy*, vol. 282, p. 116177, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261920315804

[24] T. Guo, Z. Xu, X. Yao, H. Chen, K. Aberer, and K. Funaya, "Robust online time series prediction with recurrent neural networks," in *2016 IEEE international conference on data science and advanced analytics (DSAA)*. Ieee, 2016, pp. 816–825.

[25] H. Yang, Z. Pan, Q. Tao *et al.*, "Robust and adaptive online time series prediction with long short-term memory," *Computational intelligence and neuroscience*, vol. 2017, 2017.

[26] N. Zucchet, R. Meier, S. Schug, A. Mujika, and J. Sacramento, "Online learning of long range dependencies," *arXiv preprint arXiv:2305.15947*, 2023.

[27] S. Timotheou, "The random neural network: a survey," *The computer journal*, vol. 53, no. 3, pp. 251–267, 2010.

[28] M. Nakip, B. C. Gül, and E. Gelenbe, "Decentralized online federated g-network learning for lightweight intrusion detection," in *2023 31st International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2023, pp. 1–8.

[29] M. Nakıp and E. Gelenbe, "Online self-supervised learning in machine learning intrusion detection for the internet of things," *arXiv preprint arXiv:2306.13030*, 2023.

[30] E. Gelenbe and M. Nakip, "Real-time cyberattack detection with offline and online learning," in *2023 IEEE 29th International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 2023, pp. 1–6.

[31] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[32] S. Sotirov, "Modelling the backpropagation algorithm of the elman neural network by a generalized net," in *Proceedings of the 13th International Workshop on Generalized Nets*, 2012, pp. 49–55.

[33] G. Ren, Y. Cao, S. Wen, T. Huang, and Z. Zeng, "A modified elman neural network with a new learning rate scheme," *Neurocomputing*, vol. 286, pp. 11–18, 2018.

[34] E. Gelenbe, "Stability of the random neural network model," *Neural computation*, vol. 2, no. 2, pp. 239–247, 1990.

[35] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin *et al.*, "Ad click prediction: a view from the trenches," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 1222–1230.

[36] H. Do Kim, "Predicting product demands for large-scale chain stores with ftrl-proximal linear regression," *JOURNAL OF ADVANCED INFORMATION TECHNOLOGY AND CONVERGENCE*, vol. 7, no. 2, pp. 35–42, 2017.

[37] N. A. Huynh, W. K. Ng, and K. Ariyapala, "A new adaptive learning algorithm and its application to online malware detection," in *Discovery Science: 20th International Conference, DS 2017, Kyoto, Japan, October 15–17, 2017, Proceedings 20*. Springer, 2017, pp. 18–32.

[38] E. Department for Business and I. Strategy. (2021) Weekly road fuel prices. [Online]. Available: https://data.world/makeovermonday/2020w17-weekly-road-fuel-prices

[39] berkeleyearth.org. (2021) Global climate change data. [Online]. Available: https://data.world/data-society/global-climate-change-data

[40] N. Kommenda. (2021) Britain's power sources. [Online]. Available: https://data.world/makeovermonday/2019w32