Recurrent Neural Network for Web Services Performance Forecasting, Ranking and Regression Testing

Muhammad Hasnain[a], Muhammad Fermi Pasha[b], Chern Hong Lim[c], and Imran Ghan[d]
[a,b,c]MONASH University, Malaysia
Email: {muhammad.malik1,muhammad.fermipasha,Lim.ChernHong}@monash.edu
[d]Indiana University of Pennsylvania, USA
Email: imransaieen@gmail.com

***Abstract*—Accurate estimation of web services performance, which is critical to ensure the consumers satisfaction on web services is still a challenging task due to the dynamic, and personalized requirements of different individuals. Efficient estimation of web services performance can lead to a better ranking of web services. Regression testing is then performed on the ranked web services to ensure that existing functionality of the web services is not impacted through evolution in the web services. Soft computing techniques are highly resource consuming, and more complex for practitioners. Moreover, they show complex approximation with a low propagation, which can be improved by using the advanced deep neural networks. Previously proposed web services performance estimation and analysis have been never considered from the deep neural network. To address the problem of efficient estimation of web services performance, gated recurrent unit (GRU) has been proposed with the use of time slice quality of service (QoS) data of web services. The GRU model can analyze QoS values obtained from different sets of users in different timestamps. The proposed approach has been evaluated on the web services dataset and comparison indicates that the proposed approach shows the better prediction and estimation than the state of the art approaches.**

***Index Terms*—web services, performance prediction, GRU model, quality of services**

I. Introduction

Web services paradigm is evolutionary and requires optimized testing to ensure that web services users have satisfaction on their invoked web services. Regression testing addresses the concerns produced from web services evolution. It aims at identifying the faults of modified web services [1]. Many of the existing regression testing approaches [2-3] rely on the code coverage and fault detection strategies. Code coverage and fault detection approaches render the path of maximal code coverage as well as the minimal cost can address the code problems through the test coverage optimization in the conventional software systems [4-5]. For frequent web services update, the slice based regression testing approach can fix the functionality issue with the help of software patches [6]. However, code or slice based regression testing approaches cannot be used for the web services which have been integrated by a third party web service. For instance, code unavailability of a partners web service makes above-mentioned regression testing approaches ineffective as the test coverage criteria cannot be met without code [7]. Consequently, researchers have focused on QoS features published by service providers or collected by end users [8]. However, values published by web service providers cannot be exactly same to those obtained at users' end. Moreover, QoS values in case of correlated or composite web services do not necessarily provide maximum QoS values to the end users [9]. There are certain factors which can affect in creating the divergence between the guaranteed values, and QoS values obtained by users.

Constant and unpredicted changes in performance of web services occur due to network and other contributing factors. To measure the user-perceived QoS online transactions, usually throughput and response time performance metrics are used. In [10] it has been reported that the response time is a key metric used to measure users satisfaction in web services. Therefore, customers seek quality web services which meet their business requirements, otherwise they reject web services with response time as exceeds the acceptable threshold. Moreover, definitions of response time and throughput have been given as follows:

"Response time comprises the period of time between entry of a request by a customer and completion of processing for this request while throughput is the amount of data moved successfully from one place to another in a given time period" [10].

Both the performance measuring metrics such as response time and throughput have been widely used for the selection of web services [11]. The value of each of two performance metrics is measured externally by calculating the response time in (sec), and throughput as data moved in (kbs) of a web service. Moreover, instrumentation can be internally used to measure the timing and number of events.

Performance values attained at users' end should match to a service level agreement (SLA) document [12]. Matching between the users' end attained values indicate that how a user is satisfied with the web service. Likewise, closer satisfaction and the original quality metrics values may increase the satisfaction of services users. A typical web services user prefers to select web services with the best (QoS) performance

[13]. However, web services which show a low throughput value and a high response time value cannot be ignored for future use. For instance, an institutional web service even with the changing performance values is necessarily accessed by users. This may be the same for a number of other web services from different domains. The QoS performance can be achieved by the computation of historical QoS data to forecast their performance and rank web services for further regression testing. Regression testing can be instituted for web services with the low QoS forecast before the web services with a high QoS forecast.

The adoption of deep learning (DL) in recent researches led to use deep structures in a number of novel works [14]. Deep learning approaches have been widely used for object detection, object classification, and image segmentation. Moreover, the applications of DL include the webpage aesthetics quantification and software fault detection with the help of convolutional neural networks (CNNs) [15], and deep neural networks (DNNs) [16], respectively. The DL technology has become popular due to its architecture which can help to design more updating sets of features. Moreover, the performance of DL architecture at execution time is higher than the traditional machine learning models [14].

To overcome web services performance forecasting and ranking issue, we propose to use a variant of sequence DL model known as gated recurrent unit GRU on our collected QoS dataset of web services. With the increasing amount of published web services, users require to select those web services which have better forecasted performance based on their historical invocation records. Web services performance prediction can help us in conducting the regression testing. Moreover, the web services regression testing can be performed on the predicted score of each web service. A web service with a high predicted performance score can undergo regression testing after the regression testing of web service with a low predicted performance score and so on. Performance metrics are taken into consideration for the proposed method for the 30 time slices after the web services have been published. We use throughput quality metric values as inputs to keep the problem definition simple, and applicable to other services and web applications. The main contributions of this paper are given as follow:

1) To overcome web services performance prediction and testing issue, we prorpose applications of RNN, particularly GRU models which are trained and evaluated on the time series throughput information.
2) A rank sum web services (RSws) method is defined to rank web services from GRU models using the predicted performance of web services datasets.
3) Performance prediction comparison of GRU models is performaed with the SimpleRNN, and LSTM models.
4) Web services are ranked using the RSws score of each web service, and a web service with the least RSws score is tested before the web services with a highest RSws score. Furthermore, we identify faults from test case execution from our proposed assertion analysis.

Section II of this paper presents literature review of research works which are related to this study. Section III gives description of proposed methodology for this work. Section IV is about results and discussion, and section V concludes the study.

II. Literature Review

In this section, we briefly discuss, the web services selection and deep neural networks (DNNs) based classification approaches.

A. *Web services selection and ranking*

A hybrid multi-criteria decision method MCDM [17] has been recently proposed with the consideration of multiple criteria for the selection of web services providers. Both relation and interdependencies between criteria have never been used for the selection of web services. The proposed hybrid-MCDM approach is feasible for the selection of web services providers. Web services selection is based on the performance of the vendor which means that a web service published by a well-known vendor shows a higher selection by users. This hybrid-MCDM is helpful for the selection of cloud web services because historical QoS information is not available for web services users.

Relevancy Function [18] for web services selection is mainly based on the users request. Relevancy value is determined by using quality metrics. A web services with a high relevancy values is ranked at the top of web services. Trust model [19] is proposed to optimize the web services selection through the trust rate of services providers. Users preference of a service is applied to calculate the trust rate of a service provider. Subsequently service selection approaches [18-19] mainly deal with the selection of web services from services providers aspects. Dynamic service selection can be improved by considering the correlation among QoS attributes. Both user defined QoS restraints, and inter-services correlation have been combined together to propose a model [20] which makes a tradeoff between minimizing the penalty of breach of a contract and maximizing the QoS restraints.

GTrust model [21] involves the functionality coverage which can be given by a web service to a service user. Moreover, the concept of degree of dependence represents a relationship between services in a group. For the selection of a web service group, this model is appropriate in the given scenarios. However, trust calculation of individual groups can be done in future works.

Users satisfaction and web services quality are closely related to each other in web services paradigm. A primitive research work carried out by Mohanty et al. [22], is mainly focused on the selection of web services. However, dynamic business environment has adverse effects on the users satisfaction on quality of web services.

Most of the above-mentioned studies [17-19] discuss the web services selection regarding the trust and reputation of services providers. However, trust calculated from a web service providers perspective is less efficient as a number of factors involve in determining trust of web services providers. For instance, developers reputation can be used to select the web services. Although, requirement-based regression testing techniques use the developers priority score for each requirement along with other factors. Therefore, a research work can be carried out to include the reputation and skills, experience, and number of completed projects for a software development team.

B. Deep Learning based web services evaluation

Deep Learning-Based Code Readability Model (DeepCRM) [23] involves multiple ConvNets which are skillful from different perspectives. Each ConvNet has a feature learning network that contains a convolutional layer with numerous filters, such as ReLU function and a max-pooling layer. Moreover, a classification network which is composed of fully-connected layers leads to two-way Soft-max classifier. DeepCRM outperformed many state of the art classification models. DeepCRM can be extended with the new features of software.

Deep Hybrid Services Recommendation (DHSR) [24] is composed of collaborative filtering and component contents. Feed-forward neural networks represent these two components. DHSR captures the complex interaction between web services and mashups. Description of services tags along with the mashups have been found crucial to recommend web services. Semantic similarities between services and mashups are extracted with the help of collaborative filtering.

CL-ROP [25] in the combination of Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM) is proposed to predict the reliability of web services. The former component CNN extracts the features and latter component LSTM is aimed to solve the problem of gradient diffusion which arises from long and lengthy processing of a recurrent neural network (RNN). This model has potential to be used on the dataset with the multiple quality attributes of web services.

Deep belief network (DBN) [26] is evaluated on WEBSPAM-UK2007 dataset. DBN outperformed SVM and RF as shallow classifiers. Web spam classification accuracy is approved by DBN to a certain extent. Complexity in the proposal of DBN is a big limitation in the context of a high time. This can be addressed by extended work on the combination of DBN classifier with meta-classifiers by removing the incompatible parts for better classification of web spams.

Long short-term memory (LSTM) model is a type of RNN model where the hidden layers are treated as memory unit. It is capable to correlate with time series in both long and short terms [27]. The application of RNN has been studied in [28] where software reliability was investigated regarding software failure data patterns. A feedforward RNN model showed a better prediction than the artificial neural networks. In a recently published work [29] Layered RNN (L-RNN) has been employed with the feature selection to efficiently improve the fault detection. L-RNN outperformed the ANN, Nave Bayes (NB), and logistic regression (LR) classification and prediction techniques in term of area under curve. Moreover, the proposed L-RNN model obtained the excellent classification based on AUC results. RNN models have one of the important limitations, which include gradient vanishing problem [30] that is not observed for a small number of unfolded time steps. For multiple time scales, feedback analysis cannot be precisely analyzed by using RNN models. To overcome, this issue, gated recurrent units (GRU) models have been proposed to prevent the gradient vanishing. Both, the short and long term dependencies from sequences can be efficiently learnt by GRU layer with the reset and update gates.

Deep learning models have potential to apply or extend them for various software artefacts and phases. For instance, graphical user interface can be made more user-friendly with the applications of DL models for better selection of objects to place them on webpages. Moreover, DL models can be applied at software development and testing phases for better selection of development and testing teams through forecasting their performance. Evolutionary development in deep learning is simplifying the process of object identification, performance forecasting, and classification. Optimization in improving the computation speed with a less number of layers can be adopted to solve the web services performance forecasting, and ranking problems for efficient regression testing. Hence, it seems that there is not study which is aimed at providing application of GRU model on web services' performance forecasting, and regression testing. We propose a methodology to execute the GRU model application with the optimized hyper parameters tuning.

III. Methodology

To predict web services performance accurately, the forecasting model GRU has been proposed in this section, as shown in Figure 1. For the evaluation, and performance comparison of GRU model, we parallelly run LSTM, and SimpleRNN models. The proposed methodology comprises of four phases as given in the following.

We perform GRU implementation to build a network for forecasting the performance of web services. Since long the problem of web services performance forecasting has been remained unaddressed. Many studies attempted to address the selection of web services, and GRU model-based application has never been considered. Therefore, we proposed to use sequential input data of web services and single-output model (many to one) as shown in the results and discussion section. The main input of the model is throughput metric value in a sequence from different web services users.

A. Data preparation

The first phase in our proposed methodology is given with the short description of web services datasets, and their preparation as given follows:

1) Web services dataset: We prepared web services dataset to forecast the performance of web services. The dataset was collected by using the users' load on web services for 30 time slices. The chosen five web services (WS1-WS5) have been

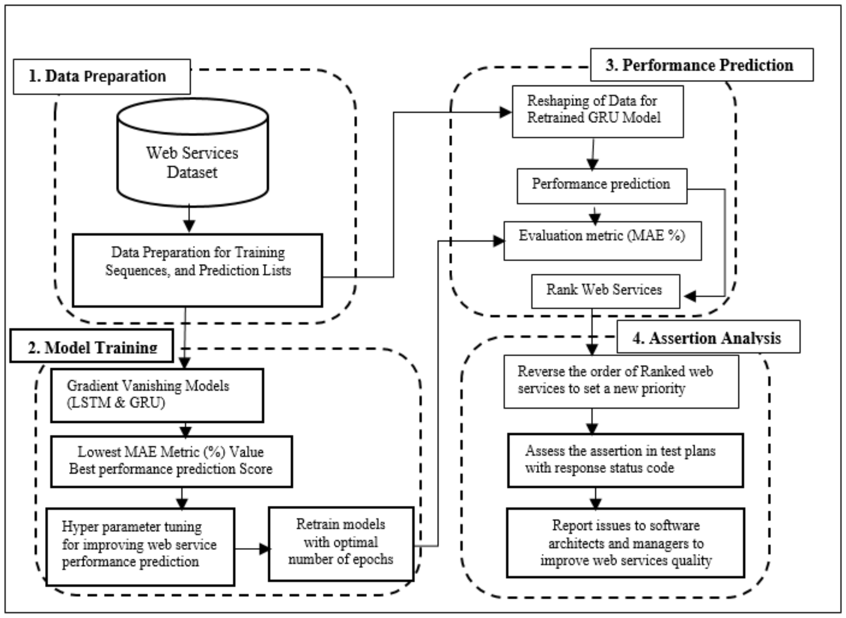
1. Data Preparation
Web Services Dataset
Data Preparation for Training Sequences, and Prediction Lists
2. Model Training
Gradient Vanishing Models (LSTM & GRU)
Lowest MAE Metric (%) Value Best performance prediction Score
Hyper parameter tuning for improving web service performance prediction
Retrain models with optimal number of epochs
3. Performance Prediction
Reshaping of Data for Retrained GRU Model
Performance prediction
Evaluation metric (MAE %)
Rank Web Services
4. Assertion Analysis
Reverse the order of Ranked web services to set a new priority
Assess the assertion in test plans with response status code
Report issues to software architects and managers to improve web services quality

Fig. 1. Proposed methodology

used in a published work [33] their metadata is accessible as WSDrean dataset from Github repository. These five chosen web services are randomly selected. We can use other web services to evaluate the efficiency of the proposed research methodology in the future work. The web services datasets were collected in Postgraduate Lab at Monash University School of IT on 17/05/2019 to 16/6/2019 for ranking of web services. We prepared dataset in 30 time slices under two different users' scenarios. This is because, we require to predict the performance of web services under changing load of users. However, number of time slices, and scenarious can be increased. For instance, two users scenarios include 100, and 200 users. To simulate the data collection, we executed load testing to determine the behavior of web services in terms of performance metrics. Hypertext transfer protocol (HTTP) GET method was used to make a request for information located on the web servers. We report response time, throughput, and latency metrics values in our web services datasets.

2) Data Usage: We use 30 time slices data into input arrays for training and testing the GRU model as well as other RNN models (LSTM, and SimpleRNN). We split the data to train and test the GRU model. Therefore, we use 90% data for training and rest of the 10% data for testing and evaluation of the model.

B. Model training

The second phase is focused on setting a criteria to train our model and hyper parameters tuning with the objectives of achieving a high accuracy metric value. This phase also shows the selection of a best model which has better efficiency to address the gradient vanishing issue.

Fig. 2 is the illustration of training and testing data of web services datasets. Time slice 1 to time slice 27 data is proposed to be used for training RNN models. Training dataset (90% of total throughput records) is used for the learning process of RNN models while testing dataset (10% of total throughput records) is used to evaluate the performance of models.

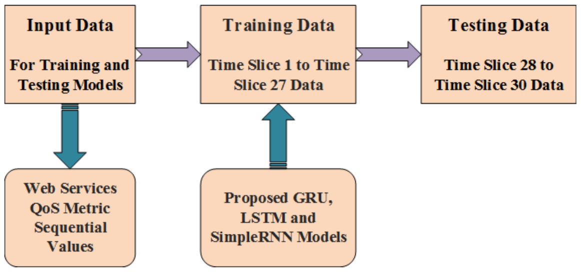
Input Data
For Training and Testing Models
Training Data
Time Slice 1 to Time Slice 27 Data
Testing Data
Time Slice 28 to Time Slice 30 Data
Web Services QoS Metric Sequential Values
Proposed GRU, LSTM and SimpleRNN Models

Fig. 2. Proposed models training and testing

1) Gradient vanishing problem: RNN models are specific to process the sequential or historical data. However, SimpleRNN model suffers from the short term memory. If sequence of input information is larger, they are inefficient to carry the information from the earlier to latter ones. The web services invocation record may come from multiple users in a long sequence, and requires an efficient memory handling mechanism to forward it for testing after performance prediction. If RNN forgets the performance value of a web service regarding the high or low values, its usefulness decreases performing the testing for identification of performance issues. For example, if a web service is accessed by users, and gives low performance value, we cannot forget this one to improve quality of a web service. Memory should be used to keep the performance of a web service, and forward the information to generate test case by using regression testing. To overcome the memory issue, various long-term memory cells have been introduced. Two most important of them are given as follows:

LSTM: To solve the vanishing gradient problem LSTM model has been proposed in [31]. A LSTM model contains three gates namely, input gate, forget gate, and output gate. The purpose of an input gate is to specify that which incoming input value is stored in the upcoming state. The forget gate is used to decide which of the previous state information is no longer stored. The output gate identifies that which of the information is sent out from the new state.

GRU: Cho et al. [32], originally proposed GRU, and is considered as a variant of the LSTM with a very simplified architecture. A GRU model comprises of reset gate r and update gate z. The purpose of update gate is to decide which portion of a hidden state h_t is to be updated with the new candidate hidden state c_t. On the other hand, reset gate is proposed with the aim to determine which portion of new state is to be ignored. Moreover, GRU has been demonstrated with the fast computation, lower complexity, and equivalent ability for learning long dependency between various time steps as compared to LSTM.

GRU with the forward pass has been given as

$$z_t = \sigma(W_{xz}\,X_t + U_{hz}\,h_{t-1} + b_z) \tag{1}$$

$$r_t = \sigma(W_{xr}\,X_t + U_{hr}\,h_{t-1} + b_r) \tag{2}$$

$$c_t = tanh(W_{xc}\,X_t + U_{hc}(r_t \odot h_{t-1}) + b_c) \tag{3}$$

$$h_t = (1 - z_t) \odot (h_{t-1} + z_t * c_t) \tag{4}$$

Where X_t denotes the input data as x = x_1, x_2 . . x_n, tanh represents the hyperbolic tangent function which is often used as an activation function of the candidate state, sigma represents the logistic sigmoid functions of reset and update gates. Moreover, c_t represents the candidate state, and h_t is the GRUs output. W_{xz}, W_{xr}, and W_{xc} are representing the weight matrices between input layer, and update gate, and reset gate as a candidate state, respectively. U_{hz}, U_{hr} , and U_{hc} are representing the weight matrices of cycle connections; while bz, br, and bc are representing the relevant bias vectors.

2) Lowest Mean Absolute Error (MAE) metric value and best performance prediction score: To evaluate the performance of prediction models, compile method is set with the MSE loss function with the objectives of penalizing the higher prediction errors. Moreover, we propose to use MAE metric to assess the quality of our used model with comparison to LSTM and SimpleRNN sequential models.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} abs(y_i - \lambda(x_i)) \tag{5}$$

To achieve the higher performance prediction score, we require hyper parameters tuning for above-mentioned models. We train GRU model on varying number of epochs, and choose the best one epoch regarding lowest MAE score. Based on the selection of best epoch and training of proposed GRU model, results in comparison to SimpleRNN, and LSTM models are presented in the section IV.

3) Hyperparameters' tuning: In the following we define the hyper parameters' tuning criteria for our proposed GRU model.

- To train GRU model, increasing number of epochs may increase its performance. Therefore, we train models between 100 and 1000 epochs.
- We use three activation functions in the hidden layer which include Sigmoid, tanh, and ReLU. However, after GRU training on different number of epochs, we choose ReLU as an activation function as it has better training and testing accuracies than the other activation functions. Moreover, ReLU does not show back propagation errors unlike the other activation functions.
- ReLU computation is fast as compared to other activation functions.
- Increasing the number of neurons in Dense layer may increase the efficiency of GRU model. Therefore, we tune this layer with the increasing number of neurons and find the best results with 100 neurons.

The purpose of tuning is to retain the best GRU model for the performance prediction of web services.

C. Performance prediction of web services

The proposed third phase as performance prediction is aimed to predict the performance value of each web service dataset from 30 time slices information. The third phase of the proposed methodology is explained in the following:

1) Sequential input data: In the preceding phase of proposed methodology, we retain best GRU model with high accuracy results. We required sequential input data for demonstration of performance prediction. For this, we propose to use sequential data as input from 30 time slices.

2) Performance prediction: Since we use GRU model with data from 30 time slices, output as performance prediction is received with one value. GRU model keeps them in the memory and releases the performance prediction information for further prioritization of web services.

3) Evaluation computation: To obtain the forecasted performance value of web services for the next time slice, we take output values to rank web services. Moreover, the value of accuracy metric MAE is kept for compilation of GRU model. To compare the performance of GRU model with SimpleRNN and LSTM, same hyper parameters are used.

4) Ranking web services: The output value of each web service is used to rank the web services. A web service with the high forecasted performance value does not require to be executed for regression testing before a web service with the lesser performance value. Therefore, we set the priority of each web service dataset regarding the respective forecasted performance score of web services. For the purpose of ranking web services, a rank sum web service (RSws) method is defined which gives the rank of a web service for predicted values as given in the Eq. (6).

$$RS(ws) = \frac{1}{N}\sum_{k=1}^{n} k \tag{6}$$

Where k is representing the predicted values, and N is the number of predicted values from the models.

D. Assertion Analysis

To specifiy the faults after the execution of test plans, assertions are proposed to be used for regression test case prioritization. An assertion is applied to compare the actual test results with the expected results from users' requests. To analyse the assetion results, we proposed to use response status as a metric to report the outome of test plans. To automate the detection of faults in web services, we create assertions in JMeter that tells how a ranked web service performs against the expected performance. Assertions' response gives an exact information of issues in a ranked web service. For example, a response code status indicates which faults a web service is primarily having at the moment. An assertion verifies whether a test is passed or failed. It shows 200 OK as a response status code if a test case passes without any failure. Moreover, if a test case fails, it shows different types of error codes.

IV. Results and Discussion

This section presents results and discussion on GRU models, regarding performance prediction of web services in 30 time slices. We were interested to predict the performance of web services datasets for the next time slice from GRU-models. To do so, we ran proposed GRU-100, and GRU-200 models on throughput quality metric values obtained from 100, and 200 users respectively. We examined two different number of users to reveal the changing performance behaviour of web services. Both of the GRU models have same hyperparameters. However, the proposed GRU-100, and GRU-200 models vary in their input values. For instance, GRU-100 is trained and evaluated on WS1-WS5 data of 100 users, while training and evaluation of GRU-200 is based on the data of 200 users from same web services.

The actual time slice plot of web services (WS1-WS5) performance in 1-30 time slice is shown in Fig. 3-12.

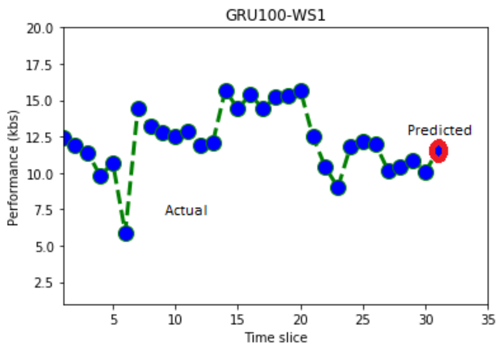

Fig. 3. GRU-100 Model for WS1 Dataset

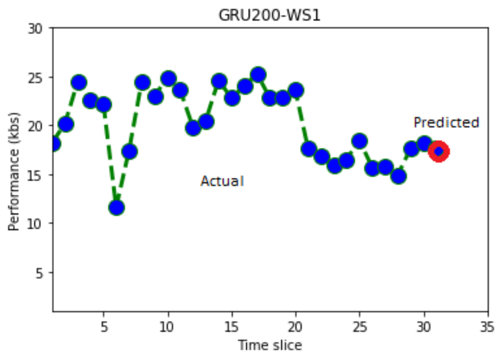

Fig. 4. GRU-200 Model for WS1 Dataset

Fig. 3-4 show the performance prediction results of WS1 from GRU-100, and GRU-200 models. We illustrate the performance prediction results for 30 time slices defined in the methodology section. Fig. 3 shows a matching of predicted performance with the actual performance in a number time slices. Fig. 4 also shows a matching of the predicted performance with the actual performance in various time slices. From these matchings, It is indicated that GRU-100 model has much better performance prediction than GRU-200 model.

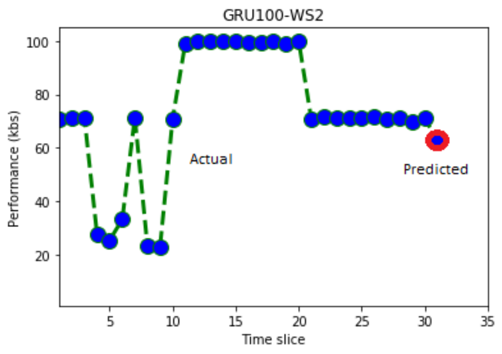

Fig. 5. GRU-100 Model for WS2 Dataset

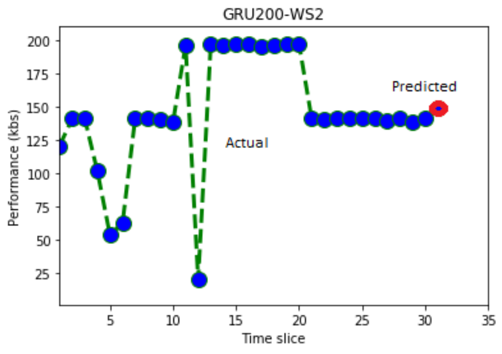

Fig. 6. GRU-200 Model for WS2 Dataset

Fig. 5-6 present the performance prediction results of WS2 from GRU-100, and GRU-200 models. Based on the actual, and predicted performance values shown in Fig. 5, and 6, there is a high volatility in the actual performance values that gives evidence about the spike detention of GRU-100, and GRU-200 models. Overall, GRU-100 has much better prediction accuracy than GRU-200 for WS2 as given in Table IV.

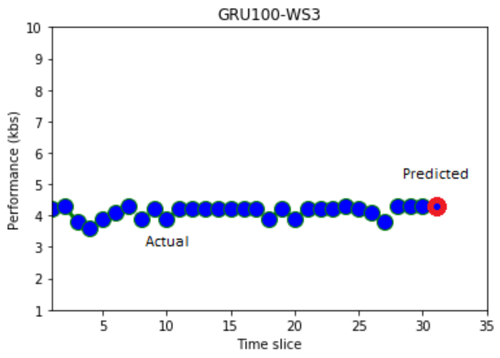

Fig. 7. GRU-100 Model for WS3 Dataset

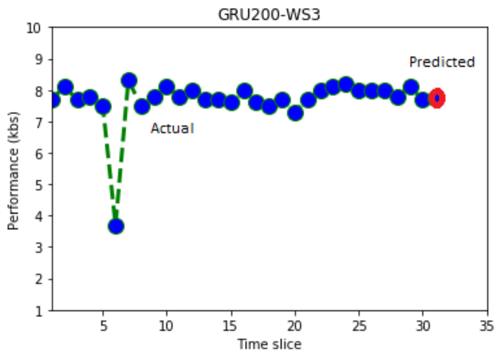

Fig. 8. GRU-200 Model for WS3 Dataset

Fig. 7-8 present the performance prediction results of WS3 from GRU-100, and GRU-200 models. Based on the actual, and predicted performance values shown in Fig. 7, it is observed that GRU-100 model performance prediction is closely matched to the actual performance values. As shown in Fig. 8, the performance prediction of GRU-200 from WS3 also matches to the actual values in the maximum number of time slices. We also find that GRU-100 has excellent prediction capability as compared to GRU-200 model from WS3 dataset.

Fig. 9-10 show the performance prediction results of WS4 from GRU-100, and GRU-200 models. We detected volatility in the actual performance of WS4 for GRU-100 model. Predicted perforamance of WS4 was found to be closely matching to the actual values as shown in Fig. 9. On the other hand, performance prediction of WS4 from GRU-200 is better than GR-100.

Fig. 11-12 present the performance prediction results of WS5 from GRU-100, and GRU-200 models. We observed one spike in the actual performance of WS5, as shown in Fig. 11. We also see three spikes in the actual performance of WS5, as shown in Fig. 12. Overall, the performance prediction of

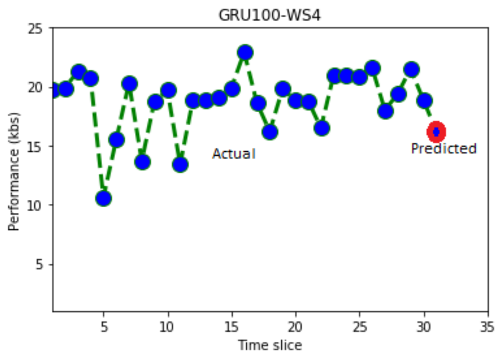

Fig. 9. GRU-100 Model for WS4 Dataset

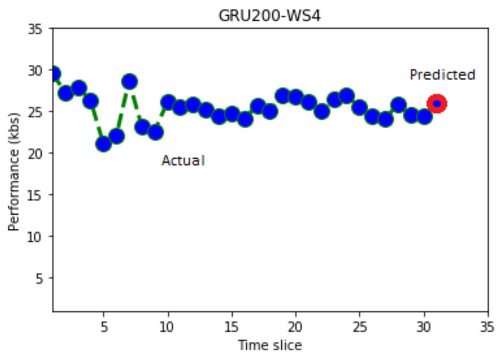

Fig. 10. GRU-200 Model for WS4 Dataset

WS5 is highly matched to the actual performance from both GRU-100, and GRU-200 models.

In Fig. 3-12, we observed the prediction results of GRU-100, and GRU-200 models from WS1-WS5 datasets. Both GRU models provide the accurate approximation to the actual performance data. We then show the ranking of web services from predicted performance of web services (WS1-WS5) datasets.

Table1 is the illustration of predicted performance results for the next time slice. Based on the next time slice predicted performance values of web services, we determine RS(ws) score of all individual datasets. The proposed GRU-100, and

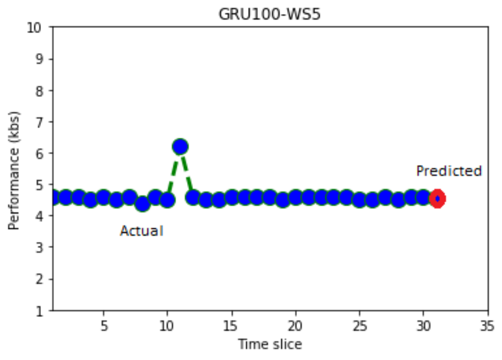

Fig. 11. GRU-100 Model for WS5 Dataset

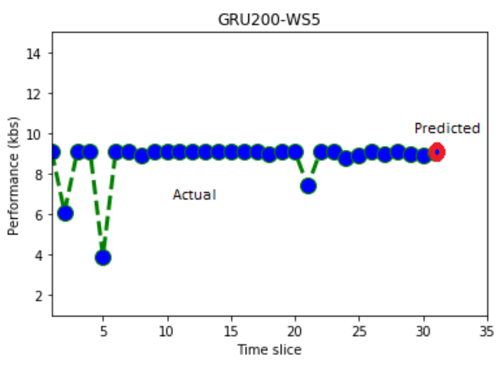

Fig. 12. GRU-200 Model for WS5 Dataset

TABLE I
PERFORMANCE PREDICTION FOR WEB SERVICES DATASETS

Model	Performance Predicted (WS1)	Performance Predicted (WS2)	Performance Predicted (WS3)	Performance Predicted (WS4)	Performance Predicted (WS5)
GRU-100	11.86	62.96	4.31	16.24	4.56
GRU-200	17.58	148.8	7.72	25.84	9.06

GRU-200 models on different datasets show various results. For instance, GRU-200 model shows better performance prediction than GRU-100 model from WS1-WS5 datasets. This is due to larger size of dataset (200 users) which is used to train the GRU-200 model in comparison with the GRU-100 model that is trained on a smaller size of dataset (100 users).

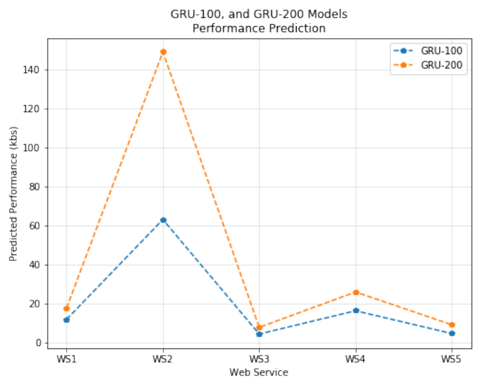

Fig. 13. GRU-100, and GRU-200 Models for web services performance prediction

Fig. 13 is the presentation of predicted performance values of WS1-WS5 from GRU-100, and GRU-200 models. Difference between predicted performance values of WS1 from GRU-100, and GRU-200 is not highly significant as shown in Fig.13. Therefore, performance of WS1 improves little while increasing the number of users. However, the predicted performance values of WS2 from GRU-100, and GRU-200 models are greater with the marginal difference. It means that performance of WS2 improves while the number of users increase. Moreover, we noticed that performance prediction of WS3 remained lower than the rest of web services. Also, difference between performance prediction of GRU-100, and GRU-200 from WS3 was found least. It indicates, WS3 performance does not improve with the increase of number of users. As displayed in Fig. 13, we observe the difference between predicted performance values of WS4 from GRU-100, and GRU-200 models. Furthermore, we knew that the predicted performance of WS4 improved while increasing the number of users. However, performance improvement of WS4 for 200 users was better than WS1, WS3, and WS5 web services. The peroformance prediction of WS5 from GRU-100, and GRU-200 models was better than the WS3 web service.

TABLE II
WEB SERVICES RANKING

Dataset	RS(ws) score	New Ranking
WS1	14.72	3
WS2	105.88	1
WS3	6.02	5
WS4	21.04	2
WS5	6.81	4

Table II is showing us the rank sum values as well as new ranking of web services. The new ranking of web services has been determined by using the proposed RS(ws) method given in Eq. (6). We have prioritized the web services by calculating RS(ws) score of each web service. From Table II results, we can set the new priority of web services regarding their calculated RS(ws) score. This priority of web services is further used in the next phase of the proposed approach.

A. Assertion analysis results

The Get method of HTTP under different scenarios with ramp up period, and loop counts is esecuted. Since we perform regression testing, we apply GET parameter to request information from web services. We achieve various assertions which indicate whether a test plan is successful or it is producing some faults. Results are reported according to the new ranking of web services as given follows:

TABLE III
TEST PLANS, AND RESPONSE CODE RESULTS

Dataset	Test Plan Scenarios (Users)	Response Code
WS1	1000	301, 404, 502, Non-HTTP
WS2	1000	200
WS3	1000	302, Non-HTTP
WS4	1000	200, 301
WS5	1000	200, Non-HTTP

Table III is the illustration of test plans and response code results of ranked web services. We have achieved various response codes with 1000 users test plan scenario. We have experienced five types of faults while accessing the web services under the same users load as shown in Table III. For web services datasets (WS1 - WS5), we experience 300, 301, 404, 502 Non-HTTP errors whereas we also experience 200 response code which indicates that the HTTP request to a web service is successfully processed. We highlight the faults identified in web services by using test plans as shown in Table III. By the execution of test plans with low to high number of users may help us in identifying the first fault in the web service.

B. Performance Comparison of GRU-100, and GRU-200 with LSTM, and SimpleRNN models

In this comparion, GRU-100, and GRU-200 are compared with the their forefather LSTM, and SimpleRNN predictors, where GRU-100, GRU-200, and their contenders have same origin called as recurrent neural networks.

TABLE IV
PREDICTION ERROR COMPARISON RESULTS OF GRU-100 MODEL

Dataset	GRU-100	LSTM	SimpleRNN
WS1	0.70%	0.79%	1.35%
WS2	4.87%	4.99%	8.81%
WS3	0.17%	0.19%	0.16%
WS4	1.38%	1.17%	2.07%
WS5	0.19%	0.19%	0.10%

Table IV is presenting accuracy comparison results of GRU-100, LSTM, and SimpleRNN from WS1-WS5 web services with the first scenario of 100 users. As shown in Table IV, the next time slice performance prediction of web services (WS1-WS5) from LSTM, and SimpleRNN models remained close to our proposed GRU-100 model. There is no big MAE (%) difference among three contenders with the typical deep architecture, but still proposed GRU-100 has better performance than others. High accuracy of GRU-100 model can be attributed to the simple structure of model with a fewer gates.

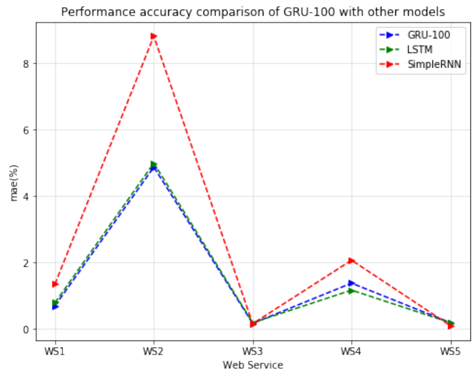

Fig. 14. Prediction Accuracy of GRU-100, LSTM, and SimpleRNN models

Fig. 14 is the illustration of prediction accuracy comparion of GRU-100 with LSTM, and SimpleRNN models. Accuracy of both GRU-100, and LSTM models is shown very close to each other from all web services datasets. On the other hand, the performance accuracy of SimpleRNN model is lower than GRU-100 model in the maximum cases of web services datasets. From Fig. 5 we observed that WS2 showed deviation in actual performance values. The SimpleRNN model showed limitation while predicting the performance of WS2 as shown in Fig.14. It is concluded that the smaller deviation in the actual values, the better prediction accuracy we obtain from SimpleRNN model. However, LSTM and our proposed GRU-100 model show better prediction accuracy for highly performance fluctuating web services.

TABLE V
PREDICTION ERROR COMPARISON RESULTS OF GRU-200

Dataset	GRU-200	LSTM	SimpleRNN
WS1	1.41%	0.93%	1.93%
WS2	8.23%	8.29%	11.35%
WS3	0.39%	0.46%	0.40%
WS4	0.82%	0.77%	1.23%
WS5	0.33%	0.53%	0.23%

Table V is presenting accuracy comparison results of GRU-200 with LSTM, and SimpleRNN models from WS1-WS5 web services with second scenario of 200 users.

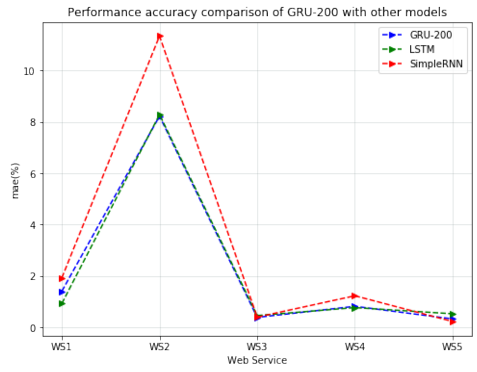

Fig. 15. Prediction Accuracy of GRU-200, LSTM, and SimpleRNN models

Fig. 15 is the demonstration of performance accuracy comparison of GRU-200 with LSTM, and SimpleRNN models. Similar to GRU-100 models' prediction accuracy, our proposed GRU-200 showed better accuracy for performance prediction of WS1-WS5 web services datasets.

High prediction accuracy of GRU-100, and GRU-200 models for (WS1-WS5) datasets indicates that both of our proposed GRU models perform better for a varying sequential data of 30 time slices. We noticed that performance of GRU-100, GRU-200, LSTM, and SimpleRNN for web services with lesser changes in their actual values does not show a high difference in prediction accuracies of above-discussed models. This verifies that SimpleRNN model still works better for the non-varying sequential data. This is why prediction accuracy of SimpleRNN model has a slight edge in results over GRU-100, GRU-200, and SimpleRNN models for WS3, and WS5 web services datasets. However, GRU-100, and GRU-200 models show a high prediction accuracy for web services (WS1, WS2, and WS4) datasets.

Overall, it is concluded that GRU-100, and GRU-200 models have better position among other used models to address the performance prediction of web services with the higher deviation in their actual performance values. Of course, as data from more time slices is added, the performance of GRU-based models will be much better than SimpleRNN and LSTM. Moreover, the ranking, and regression testing of web services based on the performance prediction from GRU-100, and GRU-200 models is highly effective in terms of reduction in computation expense. Our proposed GRU models are capable to store the hidden state information of web services performance in the continuous valued memory.

V. Conclusion and Future Implications

This paper presents the novel web services performance forecasting approach based on the implementation and application of GRU model. This work provides valuable performance prediction for ranking of web services. Moreover, GRU model with the web services performance score retaining capability in its memory has been verified on web services dataset. Based on the results, web services performance can be improved by using the ranking criteria for regression testing. Furthermore, this study illustrates that efficient memory with a fewer gates can bring more precision in performance prediction. Our study also shows that performance accuracy metric MAE% remained better for our proposed GRU-100, and GRU-200 in comparison to LSTM and SimpleRNN models. The assertion analysis of ranked web services helps in identifying the faults in the web services. This work has implications for software testers and managers to apply soft testing techniques after predicting the short term performance of web services.

References

[1] L. Mei, W. K. Chan, T. Tse, and R. G. Merkel, ”XML-manipulating test case prioritization for XML-manipulating services,” Journal of Systems and Software, vol. 84, no. 4, pp. 603-619, 2011.

[2] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, ”Prioritizing test cases for regression testing,” IEEE Transactions on software engineering, vol. 27, no. 10, pp. 929-948, 2001.

[3] B. Jiang, Z. Zhang, W. K. Chan, T. Tse, and T. Y. Chen, ”How well does test case prioritization integrate with statistical fault localization?,” Information and Software Technology, vol. 54, no. 7, pp. 739-758, 2012.

[4] A. Joseph and G. Radhamani, ”Hybrid Test Case Optimization Approach Using Genetic Algorithm With Adaptive Neuro Fuzzy Inference System for Regression Testing,” Journal of Testing and Evaluation, vol. 45, no. 6, pp. 2283-2293, 2017.

[5] Y.-D. Lin et al., ”Test coverage optimization for large code problems,” Journal of Systems and Software, vol. 85, no. 1, pp. 16-27, 2012.

[6] A. Marback, H. Do, and N. Ehresmann, ”An effective regression testing approach for php web applications,” in 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, 2012, pp. 221-230: IEEE.

[7] D. Qiu, B. Li, S. Ji, and H. Leung, ”Regression testing of web service: a systematic mapping study,” ACM Computing Surveys (CSUR), vol. 47, no. 2, p. 21, 2015.

[8] Z. Saoud, N. Faci, Z. Maamar, and D. Benslimane, ”A fuzzy-based credibility model to assess Web services trust under uncertainty,” Journal of Systems and Software, vol. 122, pp. 496-506, 2016.

[9] S. Deng, H. Wu, D. Hu, and J. L. Zhao, ”Service selection for composition with QoS correlations,” IEEE Transactions on Services Computing, vol. 9, no. 2, pp. 291-303, 2014.

[10] A. O. Ajayi, G. A. Aderounmu, H. A. Soriyan, and A. David, ”An intelligent quality of service brokering model for e-commerce,” Expert Systems with Applications, vol. 37, no. 1, pp. 816-823, 2010.

[11] J. Liu, M. Tang, Z. Zheng, X. F. Liu, and S. Lyu, ”Location-aware and personalized collaborative filtering for web service recommendation,” IEEE Transactions on Services Computing, vol. 9, no. 5, pp. 686-699, 2015.

[12] J. Heinermann and O. Kramer, ”Machine learning ensembles for wind power prediction,” Renewable Energy, vol. 89, pp. 671-679, 2016.

[13] H. A. Kadhim and H. N. Nawaf, ”Improve the Accuracy of Dirichlet Reputation System for Web Services,” in 2018 11th International Conference on Developments in eSystems Engineering (DeSE), 2018, pp. 78-82: IEEE.

[14] A. Brunetti, D. Buongiorno, G. F. Trotta, and V. Bevilacqua, ”Computer vision and deep learning techniques for pedestrian detection and tracking: A survey,” Neurocomputing, vol. 300, pp. 17-33, 2018.

[15] Q. Dou, X. S. Zheng, T. Sun, and P.-A. Heng, ”Webthetics: Quantifying webpage aesthetics with deep learning,” International Journal of Human-Computer Studies, vol. 124, pp. 56-66, 2019.

[16] W. Geng, ”Cognitive Deep Neural Networks prediction method for software fault tendency module based on Bound Particle Swarm Optimization,” Cognitive Systems Research, vol. 52, pp. 12-20, 2018.

[17] A. Al-Faifi, B. Song, M. M. Hassan, A. Alamri, and A. Gumaei, ”A hybrid multi criteria decision method for cloud service selection from Smart data,” Future Generation Computer Systems, vol. 93, pp. 43-57, 2019.

[18] M. Suchithra and M. Ramakrishnan, ”Efficient Discovery and Ranking of Web Services Using Non-functional QoS Requirements for Smart Grid Applications,” Procedia Technology, vol. 21, pp. 82-87, 2015.

[19] R. Gupta, R. Kamal, and U. Suman, ”A QoS-aware optimal selection scheme for web services with a trusted environment,” CSI transactions on ICT, vol. 3, no. 1, pp. 13-21, 2015.

[20] H. Liang and Y. Du, ”Dynamic service selection with QoS constraints and inter-service correlations using cooperative coevolution,” Future Generation Computer Systems, vol. 76, pp. 119-135, 2017.

[21] X. Su, M. Zhang, and Y. Mu, ”Trust-based group services selection in web-based service-oriented environments,” World Wide Web, vol. 19, no. 5, pp. 807-832, 2016.

[22] R. Mohanty, V. Ravi, and M. R. Patra, ”Web-services classification using intelligent techniques,” Expert Systems with Applications, vol. 37, no. 7, pp. 5484-5490, 2010.

[23] Q. Mi, J. Keung, Y. Xiao, S. Mensah, and Y. Gao, ”Improving code readability classification using convolutional neural networks,” Information and Software Technology, vol. 104, pp. 60-71, 2018.

[24] R. Xiong, J. Wang, N. Zhang, and Y. Ma, ”Deep hybrid collaborative filtering for Web service recommendation,” Expert Systems with Applications, vol. 110, pp. 191-205, 2018.

[25] H. Wang, Z. Yang, Q. Yu, T. Hong, and X. Lin, ”Online reliability time series prediction via convolutional neural network and long short term memory for service-oriented systems,” Knowledge-Based Systems, vol. 159, pp. 132-147, 2018.

[26] Y. Li, X. Nie, and R. Huang, ”Web spam classification method based on deep belief networks,” Expert Systems with Applications, vol. 96, pp. 261-270, 2018.

[27] Z. Zhao, W. Chen, X. Wu, P. C. Chen, and J. Liu, ”LSTM network: a deep learning approach for short-term traffic forecast,” IET Intelligent Transport Systems, vol. 11, no. 2, pp. 68-75, 2017.

[28] P. Roy, G. Mahapatra, P. Rani, S. Pandey, and K. Dey, ”Robust feedforward and recurrent neural network based dynamic weighted combination models for software reliability prediction,” Applied Soft Computing, vol. 22, pp. 629-637, 2014.

[29] H. Turabieh, M. Mafarja, and X. Li, ”Iterated feature selection algorithms with layered recurrent neural network for software fault prediction,” Expert Systems with Applications, vol. 122, pp. 27-42, 2019.

[30] J. Liu, C. Wu, and J. Wang, ”Gated recurrent units based neural network for time heterogeneous feedback recommendation,” Information Sciences, vol. 423, pp. 50-65, 2018.

[31] S. Hochreiter and J. Schmidhuber, ”Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[32] K. Cho et al., ”Learning phrase representations using RNN encoder-decoder for statistical machine translation,” arXiv preprint arXiv:1406.1078, 2014.

[33] Z. Zheng, Y. Zhang, and M. R. Lyu, ”Investigating QoS of real-world web services,” IEEE transactions on services computing, vol. 7, no. 1, pp. 32-39, 2012.