Evaluation of Deep Learning Models for Network Performance Prediction for Scientific Facilities

Makiya Nakashima
Texas A&M University
Commerce, TX
mnakashima@leomail.tamuc.edu

Alex Sim
Lawrence Berkeley Nat'l Laboratory
Berkeley, CA
asim@lbl.gov

Jinoh Kim
Texas A&M University
Commerce, TX
jinoh.kim@tamuc.edu

ABSTRACT

Large data transfers are getting more critical with the increasing volume of data in scientific computing. While scientific facilities manage dedicated infrastructures to support large data transfers, predicting network performance based on the historical measurement would be essential for workflow scheduling and resource allocation in the facility. In this study, we empirically evaluate deep learning (DL) models with respect to the prediction accuracy of network performance for scientific facilities, using a two-month network communication log. This paper compares a set of DL models based on Artificial Neural Network (ANN), Convolutional Neural Network (CNN), Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM), to predict average throughput as a means to estimate network performance, and shares the observations made from the extensive experiments with the results of prediction accuracy and timing complexity.

CCS CONCEPTS

• **Networks** → **Network performance analysis**; *Network measurement*; *Network monitoring*; • **Computing methodologies** → **Supervised learning by regression**.

KEYWORDS

network performance prediction, deep learning, regression, scientific computing

ACM Reference Format:
Makiya Nakashima, Alex Sim, and Jinoh Kim. 2020. Evaluation of Deep Learning Models for Network Performance Prediction for Scientific Facilities . In *3rd International Workshop on Systems and Network Telemetry and Analytics (SNTA '20), June 23, 2020, Stockholm, Sweden.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3391812.3396272

1 INTRODUCTION

Large data transfers are getting more critical with the increasing number of data-intensive applications in the big data era. This trend can easily be found from scientific computing since large data transfers among computing facilities are inherent in many scientific workflows to bring the data into the local facility for experimenting [7, 9, 12]. For instance, a recent study shows that the average file size is greater than 40 GB, which takes around 55 seconds on average to move in, for a scientific application of Advanced Light Source (ALS) [1]. Linac Coherent Light Source (LCLS) at SLAC National Accelerator Laboratory produces terabytes of data for a single experiment, which is delivered to the National Energy Research Scientific Computing Center (NERSC) computing facility to process [5].

Table 1: Summary statistics of the measured variables: (x, y) = a pair of data for January and February 2019, respectively.

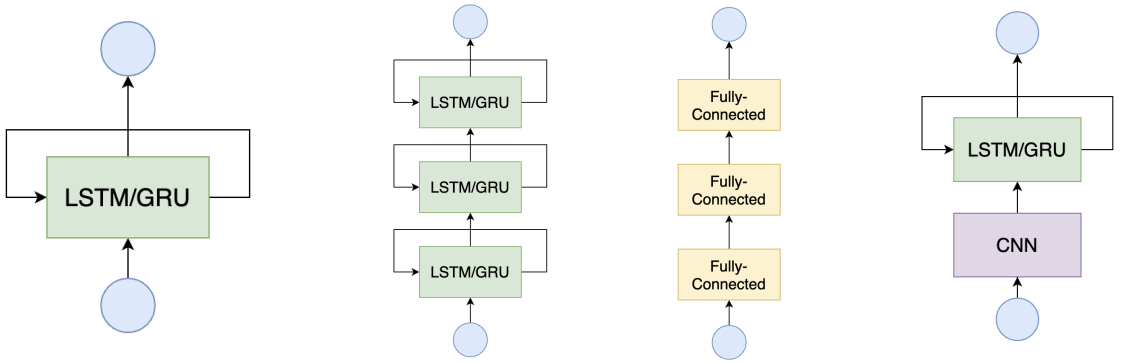
STAT	`numConn`	`aggBytes` (MB)	`avgTput` (KB/s)
Median	(18.0, 27.0)	(0.25, 0.25)	(0.25, 0.16)
Mean	(78.9, 47.1)	(1,509, 143.6)	(98.9, 36.7)
Stdev	(184.1, 52.9)	(4,952, 1,923)	(335.1, 508.0)
Max	(3495.0, 891.0)	(43,808, 107,176)	(17,872, 29,283)

To support large data transfers, scientific facilities manage dedicated infrastructures with a variety of hardware and software tools [7]. In particular, the network communication plays a key role to disseminate massive data among scientific facilities across the wide-area setting. Data transfer nodes (DTNs) are dedicated systems for data transfers in scientific facilities that facilitate data dissemination over a large-scale network [6]. Another crucial dimension for large data transfers is the accurate network performance prediction, since estimating data transfer time is vital for workflow scheduling and resource allocation. In that regard, the connection log would be a helpful resource to infer the current and future network performance, such as for change point and anomaly detection [1, 10] and for throughput and packet loss prediction [4, 5].

In this study, we empirically evaluate deep learning (DL) models with respect to the prediction accuracy of network performance for scientific facilities. We consider a simple prediction model that takes a sequence of the past observations to make a prediction for the subsequent cycle. For this purpose, we analyze a `tstat` dataset collected from DTNs at NERSC over two months (January–February 2019). `tstat` is a network monitoring tool that captures the TCP connections with a rich set of relevant information, including the number of bytes sent and received and the connection duration. Using this measurement, we compare a set of DL models, including Artificial Neural Network (ANN), Convolutional Neural Network (CNN), Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM) that have often been utilized for prediction tasks [2, 3, 11], to estimate network performance. We share our observations based on the extensive experiments with the results of prediction accuracy and timing complexity.

SNTA '20, June 23, 2020, Stockholm, Sweden

ACM ISBN 978-1-4503-7980-9/20/06.
https://doi.org/10.1145/3391812.3396272



(a) Single-layer LSTM/GRU (b) Stacked LSTM/GRU (c) Stacked ANN (d) Combination of CNN-LSTM

Figure 1: Sketch of DL models for prediction

2 DATASET AND PREDICTION MODEL

In this study, we analyze a 2-month `tstat` data collected in January–February, 2019 from the primary DTN node (DTN_1) servicing the largest number of connections at NERSC. The `tstat` tool collects TCP instrumentation data for each flow, which can be used to analyze either real-time or captured packet traces [8]. In detail, the tool constructs individual TCP connections by referring to the TCP header in the incoming and outgoing directions, and measures the transport layer statistics, such as the number of bytes/packets sent and received, the congestion window size, and the number of packets re-transmitted. A `tstat` instance consists of 107 columns to represent the associated TCP connection. NERSC collects `tstat` data to measure the impact of the network parameter settings to TCP behavior and network throughput.

Although the `tstat` data contains both incoming (i.e., from external facilities to DTNs) and outgoing (i.e., from DTNs to external facilities), we focus on analyzing *incoming* traffic since we are more interested in downloading from data sources for computing. For the purpose of network performance prediction, this study predicts *average throughput* for incoming traffic based on historical observations using DL models. Additionally, we also apply the evaluated DL models to predict *aggregated bytes* for incoming traffic.

Here is a brief description about the prediction model used in this study. The prediction takes place based on a discrete time window with size w (e.g., w = one minute). The s number of the previous observations are referenced to make a prediction for the subsequent window. Thus, the prediction of the subsequent cycle is based on the previous time interval of $(s \times w)$. The prediction function is then defined as $Pred(\cdot)$ that takes input and produces the prediction result as output. The input is a variable set $V = \{v_k | k > 0\}$ and a sequence length s defining the number of previous elements referenced for predicting the subsequent cycle. The output is a scalar variable p. Then the prediction function is defined as: $Pred(V, s) \rightarrow p$. The prediction performance is measured based on the difference between the actual measurement (m) and the predicted value (p).

By default, we assume the prediction is made by utilizing a single variable to predict the identical variable in the subsequent window (i.e., $|V| = 1$), but we also employ a combination of variables to predict to see the impact of the utilization of multiple variables (i.e., $|V| > 1$). As mentioned, we are interested in predicting (1) average incoming throughput ($avgTput$) and (2) aggregated received bytes ($aggBytes$), for each time window. Based on the basic prediction model, $avgTput$ for the subsequent window is inferred by referencing a set of $avgTput$ values observed in the past s windows, while $aggBytes$ is predicted by referring to the previous $aggBytes$ values in those past windows.

To enable this, the *tstat* data were pre-processed to construct a table maintaining window-based observations, with the following three columns for each window:

- `aggBytes`: Aggregated bytes
- `numConn`: Number of connections
- `avgTput`: Average throughput (=`aggBytes`/`numConn`)

Table. 1 provides the summary statistics for the measured variables, which shows a high degree of variations with large standard deviations for both months (i.e., January and February in 2019). This simple analysis result indicates the prediction task would not be straightforward enough due to a huge degree of variations of the measured variables over time. The focus of this study is the evaluation of DL models empirically to see their applicability, rather than defining an optimal learning model for network performance prediction.

3 DEEP LEARNING MODELS

Using the `tstat` measurement and prediction model described in the previous section, we examine several DL models for predicting the network performance for the subsequent window. Fig. 1 provides the illustration of the DL models often used for prediction, based on ANN, CNN, GRU, and LSTM structures. As can be seen Fig. 1(a), a single LSTM or GRU cell can be used for the prediction, while multiple cells can be stacked as shown in Fig. 1(b) and Fig. 1(c). It is also possible to combine heterogeneous DL models as sketched in Fig. 1(d). In this study, we examine not only homogeneous DL models but also heterogeneous models as shown in Fig. 1(d). Note that we do not make intensive optimizations for

Table 2: RMSE for predicting $avgTput$ ($V = \{avgTput\}$, Data=January)

Model	$s = 5$	$s = 15$	$s = 30$	$s = 60$
C(s)	118126	87321	125536	74340
D(s)	207915	193880	105496	207634
G(s)	58411	118167	110756	67322
L(s)	58183	108850	139787	80361
CCC(s)	195506	221347	296393	219396
DDD(s)	309117	346295	88380	68821
GGG(s)	81606	100447	163036	185395
LLL(s)	71197	133158	234786	72297

Table 3: Top-10 testing performance for predicting $avgTput$ (Data=January)

Model	Num. variables	RMSE (training)	RMSE (testing)
L(5)	1	98494	58183
G(5)	1	97292	58411
GGD(5)	1	107531	58504
G(15)	3	94028	59890
GD(60)	1	98966	61928
G(30)	3	94989	62309
LLL(5)	3	97940	62513
L(5)	3	99997	64686
G(60)	1	94740	67322
DDD(60)	1	161026	68821

individual DL models, but employ them with default settings. The following is some specific configuration information used for our experiments. For ANN, we employ linear activation function and Adam for the gradient-based optimization. For CNN, we assume a simple one dimensional model. For LSTM and GRU, we set the dropout parameters to 0.2.

We set up a naming convention to indicate individual DL models, based on a concatenation of the first letter of the DL method: 'C' for CNN, 'D' for DNN, 'G' for GRU, and 'L' for LSTM. Hence, the model in Fig. 1(c) is called DDD, while the name of the model in Fig. 1(d) would be either CL (CNN+LSTM) or CG (CNN+GRU), depending on the cell type in the upper layer. In addition, we experiment with a set of sequence lengths (i.e., how many of previous windows are referenced to make a prediction), and the model name also includes the sequence length in the form of '(s)' where s is the sequence length used. For example, 'L(5)' indicates a single layer LSTM with $s = 5$, while 'LLL(30)' does a stacked model with three LSTM layers with $s = 30$.

4 EXPERIMENTS

4.1 Experimental setting and metrics

We assume a relatively short time interval (one minute) for the window size with different sequence lengths for predicting. For example, if $s = 5$, the prediction process references the past five windows (i.e., five minutes) to predict the subsequent window. In fact, one minute is not a small amount of time in the scheduling discipline, but we observed too many windows having no connection at all if the window size is too small (i.e., less than one minute). The following is the experimental configurations:

- Normalization: standard feature scaling (0–1)
- Window size: $w = 1$ minute
- Sequence length: $s = \{5, 15, 30, 60\}$
- Training:testing = First 60% of windows for training and the rest (40%) for testing

We utilize the metric of Root Mean Squared Error (RMSE) to compare the prediction performance. We also measure timing complexity for training and testing. The measurement of timing was taken place on a dedicated machine, and we performed three runs for each setting to report statistical deviations.

4.2 Experiments with combinations of DL models

Since we have no *a priori* knowledge regarding what models work better for predicting network performance, an extensive set of DL models are considered in our experiments. Table 2 shows the initial result collected with homogeneous DL models over different sequence lengths.

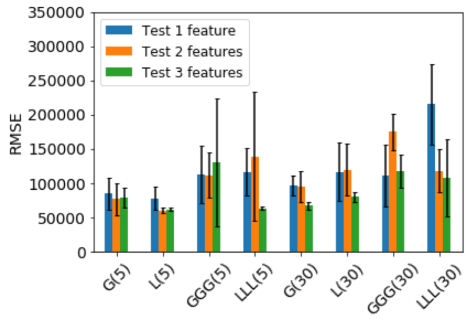
There are several interesting observations, as follows. First, using GRU or LSTM works well compared to the other structures. Second, using $s = 5$ (i.e., referencing five minutes for predicting) works better than longer sequence lengths. Using $s = 60$ (referring to past one hour for predicting) works better than $s = 15$ and $s = 30$. Another interesting observation is that, using three variables is not helpful to improve the prediction performance for CNN- and ANN-based models, while using more variables largely improve the performance for GRU- and LSTM-based models (we will discuss this again with Fig. 2 shortly).

We also constructed a variety of heterogeneous DL models by combining basic DL models. Table 3 shows the top-10 results sorted by the testing RMSE scores. From the extensive experiments, the result shows the DL models based on GRU or LSTM largely outperform other possible combinations based on CNN and ANN. With this observation, we next examine the GRU/LSTM-based models more in detail.

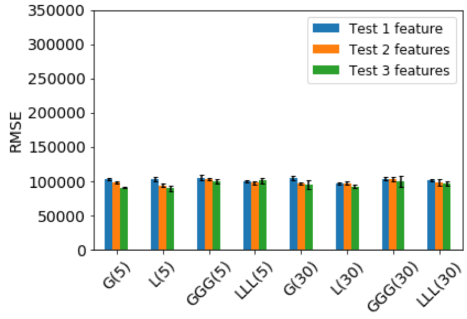
4.3 Experiments with DL models based on GRU and LSTM structures

Based on the observation above, we further evaluate the DL models based on GRU and LSTM structures. Fig. 2 shows testing results for two different months. In this experiment, we experiment with three different variable sets to predict $avgTput$: $V_1 = \{avgTput\}, V_2 = \{avgTput, numConn\}, V_3 = \{avgTput, aggBytes, numConn\}$. From the extensive experiments with a variety of combinations with different structures and parameters, the figure shows eight models that have the least RMSE measured against the January data in the *training* phase, with error bars obtained from multiple runs.

In Fig. 2, the single-layer models with $s = 5$ quite work well, yielding comparable or better results than multi-layer models or with a longer sequence length. Also, using three variables slightly works



(a) Testing RMSE for $avgTput$ (Jan)



(b) Testing RMSE for $avgTput$ (Feb)

Figure 2: Best-performed DL models for predicting $avgTput$ with the RMSE metric

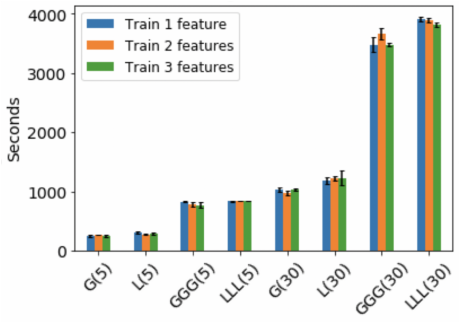


Figure 3: Learning time complexity for predicting $avgTput$ (Data=January)

consistently compared to the use of the less number of features. The testing errors are quite similar between the two datasets.

Fig. 3 compares the timing complexity for constructing learning models. Intuitively, using a smaller number of cells is beneficial for reducing the amount of time for learning data. Also, using a smaller sequence length would require a less amount of time for executing. The experimental results confirm these intuitions. However, we did not observe significant impacts of the number of variables, and we conjecture that using 1–3 variables would not be a big deal in considering high-dimensionality in practice.

5 CONCLUSIONS

In this study, we evaluated deep learning models with respect to the prediction accuracy of network performance for scientific facilities. We established a set of DL models based on ANN, CNN, GRU, LSTM, and combined DL models, to predict *average throughput* as a means to estimate network performance. From the extensive experiments, our observations show that using recurrent DL models (based on GRU or LSTM) work better than non-recurrent models (based on CNN and ANN). We also observed GRU models exhibit smaller relative differences than LSTM-based models. Overall, a simple model with a single layer and a relatively small sequence length would have some benefits, given the significantly high timing complexity for complicated models.

This work is an ongoing work and still in the initial stage. As shown in the results with relative difference, the models show high error rates. This prediction work is intrinsically challenging due to a significant degree of variations over time with a mixture of control and data channels. We plan to thoroughly analyze data channels to focus on predicting data transfer performance.

ACKNOWLEDGMENTS

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, and also used resources of the National Energy Research Scientific Computing Center (NERSC).

REFERENCES

[1] Cecilia Dao, Xinyu Liu, Alex Sim, Craig Tull, and Kesheng Wu. 2018. Modeling data transfers: change point and anomaly detection. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1589–1594.

[2] Rui Fu, Zuo Zhang, and Li Li. 2016. Using LSTM and GRU neural network methods for traffic flow prediction. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. IEEE, 324–328.

[3] Rui Fukuoka, Hiroshi Suzuki, Takahiro Kitajima, Akinobu Kuwahara, and Takashi Yasuno. 2018. Wind Speed Prediction Model Using LSTM and 1D-CNN. *Journal of Signal Processing* 22, 4 (2018), 207–210.

[4] Anna Giannakou, Dipankar Dwivedi, and Sean Peisert. 2020. A machine learning approach for packet loss prediction in science flows. *Future Generation Computer Systems* 102 (2020), 190–197.

[5] Mengtian Jin, Youkow Homma, Alex Sim, Wilko Kroeger, and Kesheng Wu. 2019. Performance Prediction for Data Transfers in LCLS Workflow. In *Proceedings of the ACM Workshop on Systems and Network Telemetry and Analytics*. 37–44.

[6] Rajkumar Kettimuthu, Zhengchun Liu, Ian Foster, Peter H Beckman, Alex Sim, Kesheng Wu, Wei-keng Liao, Qiao Kang, Ankit Agrawal, and Alok Choudhary. 2018. Towards autonomic science infrastructure: architecture, limitations, and open issues. In *Proceedings of the 1st International Workshop on Autonomous Infrastructure for Science*. 1–9.

[7] Zhengchun Liu, Rajkumar Kettimuthu, Ian T Foster, and Yuanlai Liu. 2018. A Comprehensive Study of Wide Area Data Movement at a Scientific Computing Facility.. In *ICDCS*. 1604–1611.

[8] M. Mellia, R. Lo Cigno, and F. Neri. 2005. Measuring IP and TCP Behavior on Edge Nodes with Tstat. *Comput. Netw.* 47, 1 (Jan. 2005), 1–21.

[9] Daniel A Reed and Jack Dongarra. 2015. Exascale computing and big data. *Commun. ACM* 58, 7 (2015), 56–68.

[10] Astha Syal, Alina Lazar, Jinoh Kim, Alex Sim, and Kesheng Wu. 2019. Automatic detection of network traffic anomalies and changes. In *Proceedings of the ACM Workshop on Systems and Network Telemetry and Analytics*. 3–10.

[11] Peter T Yamak, Li Yujian, and Pius K Gadosey. 2019. A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting. In *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*. 49–55.

[12] Zhao Zhang, Kyle Barbary, Frank Austin Nothaft, Evan Sparks, Oliver Zahn, Michael J Franklin, David A Patterson, and Saul Perlmutter. [n.d.]. Scientific computing meets big data technology: An astronomy use case. In *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, 918–927.