Title
NetGraf: A Collaborative Network Monitoring Stack for Network Experimental Testbeds

Authors
Kaur, Divneet
Mohammed, Bashir
Kiran, Mariam

Publication Date
2021-03-18

Peer reviewed

Dynamic Graph Neural Network for Traffic Forecasting in Wide Area Networks

Tanwi Mallick
Argonne National Laboratory
Lemont, Illinois
tmallick@anl.gov

Mariam Kiran
Lawrence Berkeley
National Laboratory
Berkeley, California
mkiran@lbl.gov

Bashir Mohammed
Lawrence Berkeley
National Laboratory
Berkeley, California
bmohammed@lbl.gov

Prasanna Balaprakash
Argonne National Laboratory
Lemont, Illinois
pbalapra@anl.gov

***Abstract*—Wide area networking infrastructures (WANs), particularly science and research WANs, are the backbone for moving large volumes of scientific data between experimental facilities and data centers. With demands growing at exponential rates, these networks are struggling to cope with large data volumes, real-time responses, and overall network performance. Network operators are increasingly looking for innovative ways to manage the limited underlying network resources. Forecasting network traffic is a critical capability for proactive resource management, congestion mitigation, and dedicated transfer provisioning. To this end, we propose a nonautoregressive graph-based neural network for multistep network traffic forecasting. Specifically, we develop a dynamic variant of diffusion convolutional recurrent neural networks to forecast traffic in research WANs. We evaluate the efficacy of our approach on real traffic from ESnet, the U.S. Department of Energy's dedicated science network. Our results show that compared to classical forecasting methods, our approach explicitly learns the dynamic nature of spatiotemporal traffic patterns, showing significant improvements in forecasting accuracy. Our technique can surpass existing statistical and deep learning approaches by achieving ≈20% mean absolute percentage error for multiple hours of forecasts despite dynamic network traffic settings.**

I. Introduction

Large scientific experimental facilities, with high-speed data production rates, are bringing enormous data movement and transfer challenges to the underlying network infrastructure that supports distributed science workflows. With this relentless growth, there is a need to develop proactive planning, resource allocation, and provisioning methods to manage the current backbone bandwidth of research wide area networks (R-WANs), while keeping costs low [1]. One of these proactive strategies could rely on network operators' ability to forecast future traffic on the network based on the current state of the network. For example, if we can forecast the traffic for several hours in the future, we can minimize the effect of congestion by diverting flows from congested links to free or less congested links; similarly, dedicated lines can be assigned to large transfers so that they do not interfere with other data transfers. The ability to forecast congestion patterns a few hours ahead can enable new traffic management strategies with traditional network protocol design, essentially allowing one to utilize underused bandwidth more efficiently and minimize overall congestion in the network infrastructure.

Network traffic forecasting in R-WANs is a formidable task, owing to a lack of regular patterns in how users access and perform data transfers over the network [2]. Compared to Internet WANs, which have periodic patterns [3], R-WANs have random traffic spikes that are difficult to understand and anticipate. The traffic on R-WANs depends on which science experiments and devices are running and which groups are involved, and it is characterized by high-variability data transfers lasting minutes or even hours [4].

Network monitoring tools such as Simple Network Management Protocol (SNMP), sflow, and netflow [5] allow collecting traffic information on network nodes and flow transfers as time-stamped data recording gigabytes of log files (GB). Most monitoring tools collect data at 30-sec intervals, giving a fine-grained view that includes other key features such as protocols used (e.g., TCP, UDP), interfaces, source and destination IP addresses, and even flow speeds. These data sets can be leveraged for developing data-driven forecasting methods. Classical statistical time-series forecasting methods such as ARIMA and Holt-Winters have been investigated for network traffic forecasting [6], but have been less effective for R-WANs, as regular and seasonal patterns do not exist [1]. Forecasting methods based on classical ML methods such as random forest and SVM have been developed to provide per-site forecasts [7], but they fail to consider the whole network and its spatial patterns and connectivity, making them less robust with respect to the dynamic nature of the R-WAN traffic [8]. Recent improvements in data collection present new opportunities for developing deep learning (DL) methods for forecasting R-WAN traffic [9]. For example, variants of convolutional neural networks (CNNs) and long-short-term memory methods (LSTMs) have been shown to provide better accuracy than classical statistical techniques for data sets with seasonality [10]. Nevertheless, existing DL methods for network forecasting are out-of-the-box methods and are not particularly customized for the R-WAN. They do not take into account the spatial correlation of the entire network and the dynamic temporal correlation of R-WANs.

We model the R-WAN networking infrastructure as a graph, where nodes and edges correspond to sites and their connec-

978-1-7281-6251-5/20/$31.00 ©2020 IEEE

2020 IEEE International Conference on Big Data (Big Data) | 978-1-7281-6251-5/20/$31.00 ©2020 IEEE | DOI: 10.1109/BIGDATA50022.2020.9512748

tivity, modeling network traffic as time series. We propose a nonautoregressive graph neural network approach to forecast traffic for multiple time steps ahead. Specifically, we develop a dynamic diffusion convolutional recurrent neural network (DDCRNN) that considers data movement as a diffusion process from one node to another based on connectivity. The spatial and temporal correlations are learned through graph diffusion convolution and recurrent units, respectively. Consequently, a single model is trained and used to forecast traffic on the entire R-WAN network. The DDCRNN that we propose is based on the diffusion convolutional recurrent neural network (DCRNN), which was originally developed for highway transportation forecasting [11], [12]. The key difference between the highway traffic DCRNN and our DDCRNN is the way in which the connectivity between the nodes is considered and the data that is handled. While we see that traffic networks seem to have regular seasonality patterns in terms of time of day and weekdays, network traffic has more pseudo-random patterns, where the flows depend on users connectivity and active projects. Additionally, in DCRNN approach, the connectivity is static and computed on the basis of (driving) distance; whereas in DDCRNN, the connectivity is dynamic and computed on the basis of the current state of the network flow traffic. This approach is designed to explicitly model the dynamic nature of the R-WAN traffic. Our main contributions are as follows:

- We develop a dynamic diffusion graph recurrent neural network architecture to forecast traffic for multiple time steps on an R-WAN that is characterized by a lack of regular patterns and seasonality.
- We demonstrate the effectiveness of the proposed method on real data from the Energy Sciences Network (ESnet), a high-performance R-WAN built by the U.S. Department of Energy (DOE) to support U.S. scientific research. We show that our approach explicitly learns the dynamic nature of spatiotemporal traffic patterns in this R-WAN and outperforms existing statistical and DL methods used for traffic forecasting by achieving ≈20% mean absolute percentage error for multiple hours of forecasts.

Using real traffic data from ESnet, our model captures key patterns among sites and links, highlighting several interesting behaviors and relationships that were not previously known, such as some sites having patterns that are more amenable for forecasting than others.

II. Related Work

Wolski et al. [13] developed a TCP performance network forecasting method to help schedule computations over distributed networks. These authors utilized statistical approaches such as ARIMA, Holt-Winters and Hidden Markov, and were successful in predicting a few time-steps by modeling traffic as stochastic processes [14]. Nevertheless, these methods are dependent on finding seasonality and using it to improve predictions. Network traffic spikes randomly with sudden data transfers and lacks seasonal patterns [15], affecting the forecasting accuracy of these methods [6]. Moreover, these approaches also fail to learn long-range dependency [16].

Research in software-defined networking (SDN) promises to provide flexible solutions for building agile networks and leveraging active monitoring, prediction, and informed decision-making [17]. Google [18] used SDN to optimize link usage by doing ”what-if” scenarios to schedule transfers. Using Multi-Protocol Label Switching, advanced forwarding schemes can control and optimize flows for packet forwarding [19]. However, the optimization techniques discussed do not exploit forecasting algorithms to make decisions.

Most R-WAN network links are underutilized, especially as routing protocols use path-finding algorithms rather than accounting for current traffic utilization. In prior work, dynamic congestion-based routing algorithms have been developed to adapt as traffic changes [20]. However, these algorithms are susceptible to oscillatory behaviors and performance degradation, as shown in [21].

The European R&E network, GEANT, used LSTM models to forecast traffic on European links, but only demonstrated 15-min forecasts [21]. Similarly stacked autoencoders [9] were used for 15-, 30- and 60-minute forecasts. However, these authors did not use highly dynamic R-WANs such as ESnet and exploited relatively simpler and shorter forecasts. Recently, DCRNN implementations, without any modifications, were successfully applied to forecast congestion events in a WAN [22]. Network traffic forecasting has explored statistical and simpler ML models as seen in [6] [7] [8], but prediction accuracy is seen to suffer because of lack of seasonality in R-WAN traffic traces.

In this paper, we show that DCRNN alone is not effective, as it does not consider the dynamic spatial and temporal traffic patterns in R-WAN networks.

III. U.S. Research Network: ESnet

The DOE science network ESnet provides services to more than 50 research sites and universities, including supercomputing facilities and major scientific instruments. ESnet also connects to 140 research and commercial networks, permitting geography-free collaboration around the world.

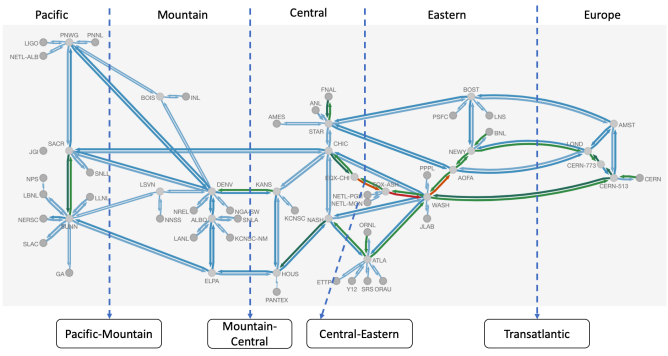


Fig. 1: Full topology of ESnet.

We use real traffic traces of two-way traffic data, collected from all of 2018, across the 48 sites in the U.S. and Europe.

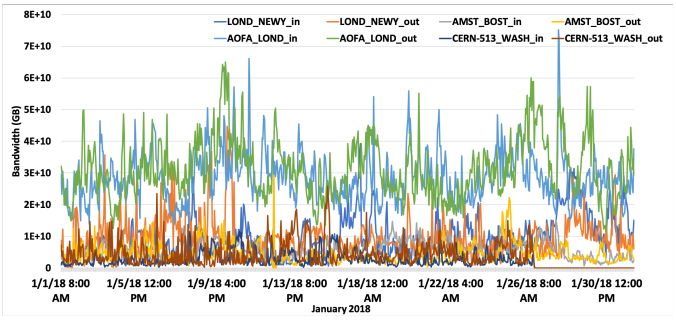


Fig. 2: Traffic patterns in January 2018 on all transatlantic links in ESnet, including London, New York, Amsterdam, Boston, AOFA, CERN, and Washington, D.C.

Figure 1 shows the full network topology of the network, divided among five time zones. All data are collected in GMT. Sample network traffic patterns on transatlantic links in ESnet appear in Figure 2, which shows the lack of temporal patterns in the network traffic.

SNMP Data Collection

Network operators monitor link capacity and data movement across routers using SNMP [23]. Table I shows a snapshot of these data during a two-way transfer between Sunnyvale and Sacramento in California.

The traffic is collected in moving GBs across router interfaces at 30-sec intervals. These data are then aggregated to 1-hour intervals and modeled as discrete observations. Aggregating traffic compounds the burst patterns [20] and reduces the model complexity, allowing faster training and predictions.

Timestamp	SACR_SUNN_in (GB)	SACR_SUNN_out (GB)
1514822400	14110930202	1025131246
1514826000	13453619303	9191557943
1514829600	12168879944	7793842045
1514833200	11231198033	7097237528
1514836800	10780847622	8048293939

TABLE I: Sample timestamped traffic trace collected from one router collecting traffic in both directions between Sacramento and Sunnyvale.

The data that we used for our study spans from 1 January 2018 to 31 December 2018, covering 48 sites with 96 traffic traces of two-way traffic.

While building hourly traffic summaries, we found that router interfaces sometimes miss recording traffic movements at specific intervals. To address this issue, we calculated the average data values for the missing points from the surrounding values to fill in the gaps. These missing values were only seen in the Washington-Chicago link during a 1-week interval in November.

IV. Dynamic Diffusion Convolution Recurrent Neural Network

We model the R-WAN as a graph $G = (V, E, A)$, where V is a set of N nodes that represents the site; E is a set of directed edges representing the connection between nodes, and $A \in R^{N \times N}$ is the weighted adjacency matrix representing the strength of connectivity between nodes. Given the historical traffic observations at each node of the graph, the goal is to learn a function f(.) that takes traffic observations for T' time steps as input to forecast the traffic for the next T time steps:

$$X(t-T'+1), ..., X(t); G \xrightarrow{\text{f(.)}} X(t+1), ..., X(t+T)$$

DDCRNN utilizes a graph-based encoder-decoder architecture that models spatial and temporal patterns through diffusion convolution operation on a graph and gated recurrent units (GRUs), respectively. Specifically, the matrix multiplication operations of the GRU cell are replaced by a diffusion convolution operation to perform graph convolution on the time-series data. A unit in the DDCRNN architecture is given by:

$$\begin{aligned} r^t &= \sigma(W_{r\bigstar G}[X_t, h_{t-1}] + b_r) \\ u^t &= \sigma(W_{u\bigstar G}[X_t, h_{t-1}] + b_u) \\ c^t &= \tanh(W_{c\bigstar G}[X_t(r_t \odot h_{t-1}] + b_c) \\ h_t &= u_t \odot h_{t-1} + (1-u_t) \odot c_t, \end{aligned} \quad (1)$$

where X_t and h_t denote the input and final state at time t, respectively; r_t, u_t, and c_t are the reset gate, update gate, and cell state at time t, respectively; $\bigstar G$ denotes the diffusion convolution operation; and W_r, W_u, and W_c are parameters of the GRU cell. The diffusion convolution operation [11] on the graph G and the input data X is defined as:

$$W_{\bigstar G} X = \sum_{d=0}^{K-1} (W_O (D_O^{-1} A)^d + W_I (D_I^{-1} A)^d) X, \quad (2)$$

where K is a maximum diffusion step; $D_O^{-1}A$ and $D_I^{-1}A$ are transition matrices of the diffusion process and the reverse one, respectively; D_O and D_I are the in-degree and out-degree diagonal matrices, respectively, and W_O and W_I are the learnable filters for the bidirectional diffusion process.

To model the dynamic nature of the R-WAN traffic, DDCRNN uses a dynamic weighted adjacency matrix estimated from the time-series data. Specifically, instead of keeping A as static, for each input sequence of T' time steps on N nodes (for both training and inference), DDCRNN computes the linear correlation coefficient values between sites that have data transfer between them and uses it in Eq. 2. Consequently, the elements in the adjacency matrix $\tilde{A}$ at time T are given by:

$$\tilde{A}_{ij} = \rho_{X_i, X_j} = \frac{cov(X_i, X_j)}{\sigma_{X_i}, \sigma_{X_j}}, \quad (3)$$

where, A_{ij} represents the edge weight between the nodes N_i and N_j; X_i and X_j are the time-series data of nodes N_i and N_j; ρ is the Pearson correlation coefficient between X_i and X_j; cov is the covariance; σ_{X_i} is the standard deviation of X_i; and σ_{X_j} is the standard deviation of X_j.

The overall architecture of DDCRNN is shown in Figure 3. There are two inputs to the DDCRNN model: 1) an adjacency matrix representing the traffic correlation among the sites of the WAN network topology and 2) the time-series data or the

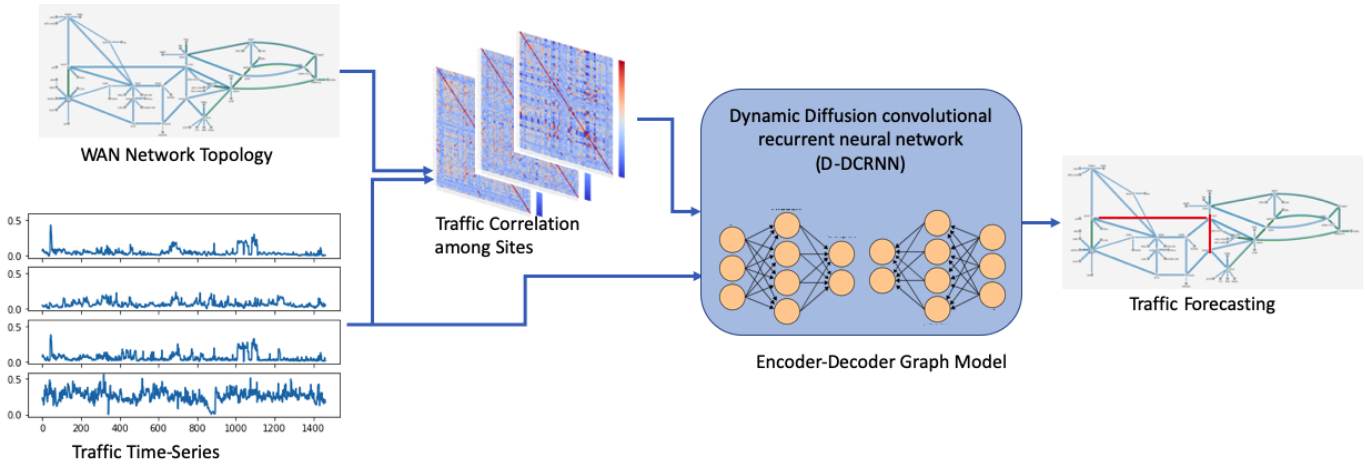


Fig. 3: DDCRNN model architecture. It takes an adjacency matrix computed from the current state of the network traffic among the sites of the WAN network topology and the traffic as a time-series at each node of the graph. The encoder-decoder deep neural network is used to forecast the network traffic for mutiple time steps.

traffic statistic at each node of the graph. The encoder of the DDCRNN network encodes the input into a fixed-length vector and passes it to the decoder. The decoder forecasts future traffic conditions. During the training phase, the time series of T' time steps on N nodes is fed in as input; the correlation matrix is computed on the time-series data and given as the weighted adjacency matrix for diffusion convolution. The encoder architecture takes the data and the decoder is used to forecast the output of the next T time steps. The learnable weights of DDCRNN are trained using a minibatch stochastic gradient, using a mean absolute error (MAE) loss function.

The DDCRNN for R-WAN traffic forecasting is based on a DCRNN that was originally developed for forecasting traffic on highway networks [11]. The network traffic forecasting is similar to highway traffic forecasting, where the traffic diffuses from one node to the other node based on the network graph connectivity. However, the key difference stems from the spatial connectivity and temporal regularity. In a highway setting, the traffic exhibits regular patterns across weekdays, peak hours, regions (i.e., distribution of vehicles going from one region to another region), and seasons. However, such patterns cannot be seen in a R-WAN network because users can initiate large data transfers at any time. Therefore, DDCRNN utilizes a dynamic adjacency matrix computed from the current state of the network traffic. During training, DDCRNN learns to diffuse traffic on the graph under different weighted adjacency matrix settings. Consequently, during inference, the trained model can perform forecasts based on the time series of T' steps and the weighted adjacency matrix computed on it.

V. Experiments

We used ESnet traffic traces for one year with a 1-hour time resolution. We grouped data as follows: from 2018-01-01 to 2018-09-13 for training (70%); from 2018-09-13 to 2018-10-20 (10%) for validation; and from 2018-10-20 to 2018-12-31 (20%) for testing. We measured the network traffic as bandwidth in GB/s. Each site has two bandwidth values: incoming and outgoing. Therefore, we mapped the 48 physical sites to a 96-node graph by considering two nodes (incoming and outgoing) for each physical site, because they have separate fiber optic links. Since data transfer varies significantly from site to site, we computed min-max scalar transformation on the training data and used it normalize bandwidth values of training, validation, and testing data.

Our DDCRNN implementation is based on the open-source DCRNN implementation [11]. We used the following hyperparameter values for DDCRNN: batch size – 64, number of epochs – 30, maximum diffusion steps – 2, number of RNN layers – 2, number of RNN units per layers – 16, max_grad_norm – 5, initial learning rate – 0.01, and learning rate decay – 0.1. These hyperparameter values were set as default for the open-source DCRNN implementation.

The DDCRNN training was performed on a single node of the Cooley GPU cluster at the Argonne Leadership Computing Facility. The node consists of two 2.4 GHz Intel Haswell E5-2620 v3 processors (6 cores per CPU, 12 cores total), one NVIDIA Tesla K80 (two GPUs per node), 384 GB of RAM per node, and 24 GB GPU RAM per node (12 GB per GPU). The software stack comprises Python 3.6.0, TensorFlow 1.3.1, NumPy 1.16.3, Pandas 0.19.2, and HDF5 1.8.17. The analysis was run on the Haswell compute nodes of the Cori supercomputer at NERSC where each node has two sockets and each socket is populated with a 2.3 GHz 16-core processor (Intel Xeon Processor E5-2698 v3) and 128 GB DDR4 2133MHz memory.

To compare the forecasting accuracy of different models, we utilized the mean absolute percentage error (MAPE) and coefficient of determination (R^2) computed on the original bandwidth scale (after applying inverse min-max scalar transformation).

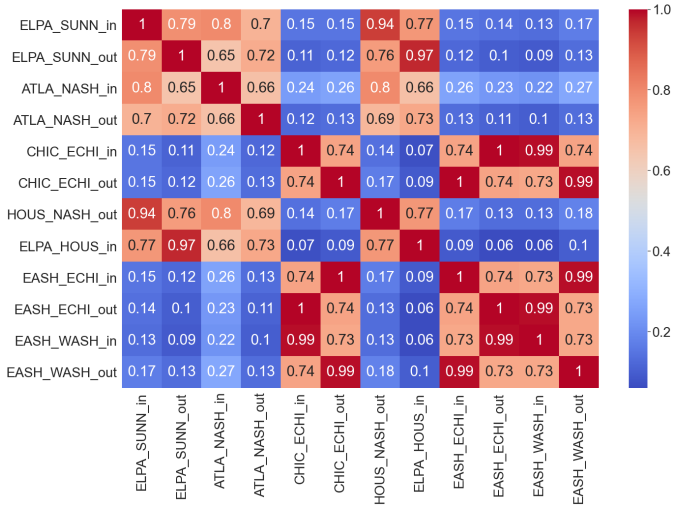


Fig. 4: Pearson correlation coefficient matrix for the 12 nodes. These nodes exhibit high correlations and were selected for the exploratory analysis.

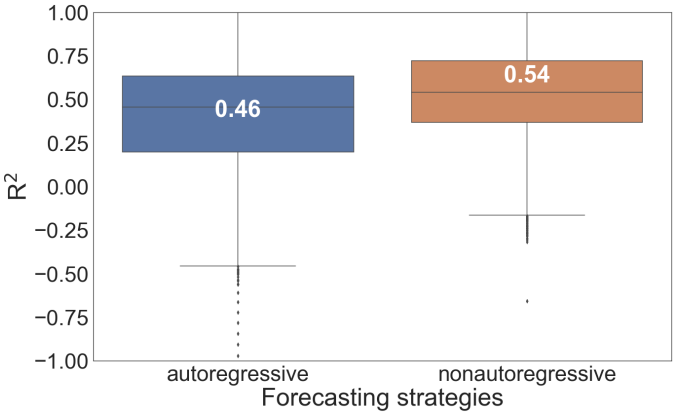


Fig. 5: Distribution of R^2 values obtained by autoregressive and nonautoregressive DDCRNN variants for the 24th-hour forecast.

A. Exploratory analysis

We conducted an exploratory analysis on DDCRNN to study the impact of input horizon duration (T'), forecasting horizon (T), and the type of autoregression on the forecasting accuracy. Given a pair of sites, we computed the Pearson correlation coefficient between their time series on the training data. We selected 12 out of 96 sites based on the strong correlation values. The correlation matrix of the 12 sites are shown in Figure 4, where _in and _out refer to incoming and outgoing nodes, respectively. The reason for selecting the subset of nodes is to avoid bias in the experimental comparison. In particular, if we use all the nodes and select the best options, then it might introduce bias in favor of the DDCRNN while comparing it with other methods. For the same reason, the forecasting accuracy for the exploratory analysis was computed on the validation data and not on the test data, thus avoiding test data leakage.

1) Impact of input horizon duration: We studied the impact of the length of the input horizon (T') on the forecasting accuracy of the DDCRNN. For training DDCRNN, we varied the length as 6, 12, 18, 24, 30, 36, 42, and 48 hours to forecast the traffic for the next 24 hours.

The R^2 and MAPE values obtained on the validation data are shown in Table II. For each node, we considered all the observed data and their corresponding predicted values from the DDCRNN on the validation set and computed R^2 and MAPE. The mean and standard deviation values were computed over 12 nodes. We observe that up to 30 hours the forecasting accuracy improves as the sequence length increases. After 30 hours, however, the improvements are not significant. Therefore, we use an input a duration of 30 hours for the rest of the experiments. Note that a larger input horizons will eventually increase the training data size and time required for training.

2) Impact of forecasting horizon: We analyzed the impact of the forecasting horizon on DDCRNN performance. Given 30 hours of input horizon, we trained the DDCRNN model to forecast 6-, 1-2, 18-, and 24-hour horizons. Note that by training the DDCRNN model for 24 hours, one can get the forecasting results for all the other horizons. However, this strategy may not be optimal for 6-, 12-, or 18-hour forecasts. Therefore, we trained DDCRNN for each forecasting horizon.

Table III shows the R^2 and MAPE values for different forecasting horizons on the validation data. From the results, as expected, we observe a trend in which the increase in the forecasting horizon decreases the accuracy. We selected 24 hours as the forecasting horizon for further study, where the DDCRNN model was trained to forecast 24 hours. However, we also analyzed the forecasting accuracy for the intermediate time intervals as described below.

3) Comparison between autoregressive and nonautoregressive DDCRNN: We compared the forecasting accuracy of autoregressive and nonautoregressive DDCRNN variants. By default, DDCRNN adopts a nonautoregressive approach to forecasting. An alternative approach is autoregressive forecasting, where DDCRNN can be trained to forecast only a one-time step, which is then given to the model recursively to obtain forecasting for 24 hours. This study was motivated by a previous work [24], where autoregressive forecasting was adopted within LSTM models for traffic forecasting.

Figure 5 shows the distribution of R^2 values obtained by the two strategies at the 24th-hour forecast. Figure 6 shows the distribution of MAPE values for different forecasting horizon intervals. The result shows that the default nonautoregressive strategy performs significantly better than the autoregressive strategy. To take a close look, we plot the forecasting accuracy (R^2) of a particular node ELPA_SUNN_in Figure 7. The results show that the forecasting accuracy of autoregressive DDCRNN decreases rapidly with an increase in forecasting time steps. This can be attributed to the error introduced at each time step, which drastically reduces accuracy for the next time

Input Horizon Duration	6 hrs	12 hrs	18 hrs	24 hrs	30 hrs	36 hrs	42 hrs	48 hrs
R^2 (μ)	0.58	0.60	0.70	0.72	**0.76**	0.69	0.71	0.77
R^2 (σ)	0.02	0.09	0.10	0.12	**0.07**	0.11	0.05	0.08
MAPE (μ)	22.71	22.60	20.90	19.50	**19.01**	22.38	20.47	20.20
MAPE (σ)	6.16	4.25	6.55	1.74	**4.13**	8.23	3.37	3.77

TABLE II: Mean (μ) and standard deviation (σ) of R^2 and MAPE values for varying input horizon durations.

Forecasting Time Horizon	R^2 μ (σ)	MAPE μ (σ)
6 hrs	0.92 (0.07)	11.31 (8.07)
12 hrs	0.82 (0.10)	16.74 (6.35)
18 hrs	0.72 (0.11)	20.61 (6.55)
24 hrs	0.76 (0.07)	19.01 (4.13)

TABLE III: Mean (μ) and standard deviation (σ) of R^2 and MAPE values for varying forecasting horizon.

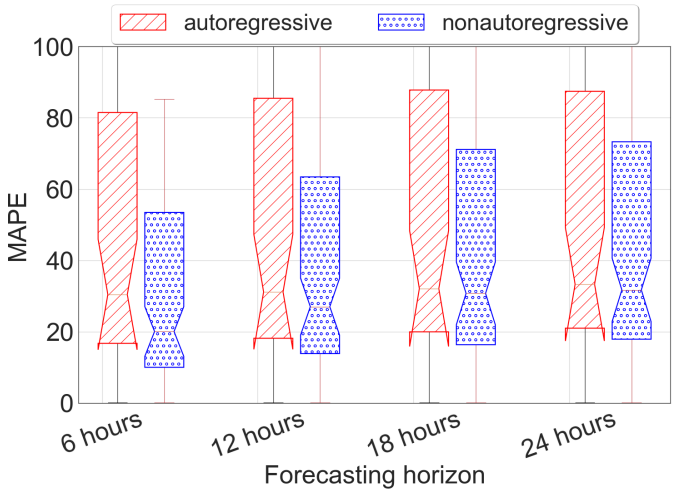
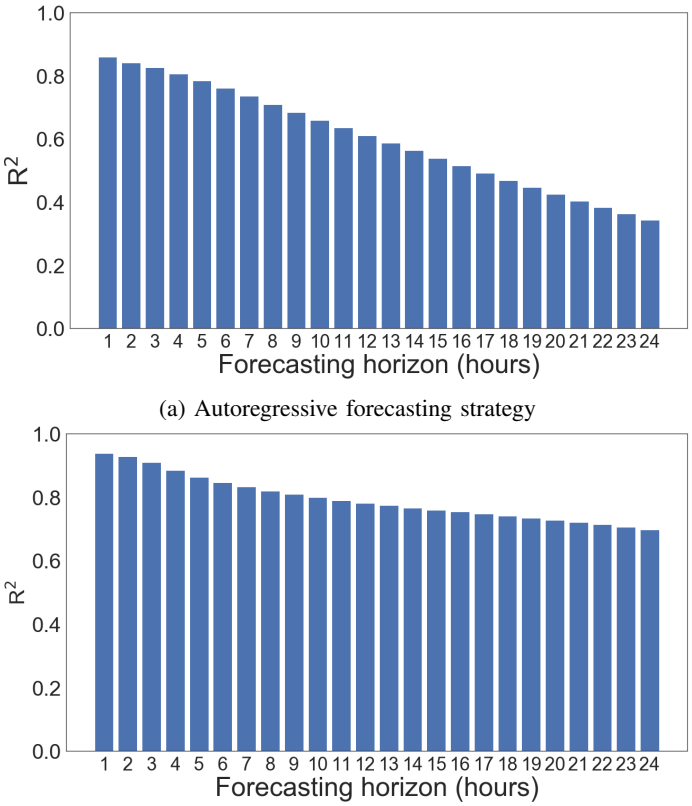


Fig. 6: Distribution of MAPE values obtained by autoregressive and nonautoregressive DDCRNN variants for different forecasting intervals.



(a) Autoregressive forecasting strategy

(b) Nonautoregressive forecasting strategy

Fig. 7: Comparison between autoregressive and nonautoregressive DDCRNN variants on ELPA_SUNN_in site.

step. This issue is less severe in the nonautoregressive strategy because it was trained on the entire forecasting horizon.

B. Comparison between DDCRNN and DCRNN

Here, we compare the forecasting accuracy of DDCRNN and DCRNN to show that our proposed dynamic approach is critical for forecasting accuracy. These two methods differ only with respect to how the weighted adjacency matrix is used in the diffusion convolution. While in the former it is dynamically computed based on the current traffic state, in the latter it is static and based on the connectivity and distance. We also note that DCRNN has been applied to network traffic forecasting previously in [22]. Both DDCRNN and DCRNN used the same hyperparameters and the same training data with 96 nodes. We used 30 hours of input horizon to forecast 24 hours of output horizon. The forecasting accuracy values were computed on the test data.

Figure 8 shows the R^2 distribution obtained by DDCRNN and DCRNN for the 24th-hour forecast. Figure 9 shows the MAPE distribution for different forecasting horizon intervals. We observe that DDCRNN achieves forecasting accuracy that is significantly better than that of DCRNN for all forecasting horizons. The superior performance of DDCRNN can be attributed to its ability to model the dynamic spatiotemporal traffic patterns; DCRNN does not have this capability because it uses the static adjacency matrix, which leads to poor forecasting accuracy. While a previous work [22] reported superior performance of the direct application of DCRNN to network traffic, we hypothesize that it was due to less dynamic traffic data with a shorter forecasting horizon.

C. Comparison of DDCRNN with other methods

Here we compare DDCRNN with other statistical, machine learning, and deep learning methods, including those previously studied for network traffic forecasting, and show that DDCRNN outperforms all these methods.

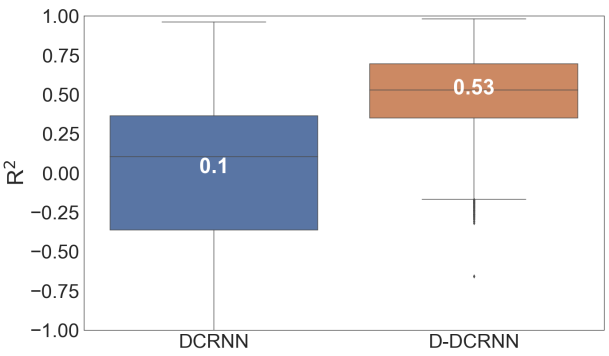


Fig. 8: Distribution of R^2 values obtained by DDCRNN and DCRNN for the 24th-hour forecast.

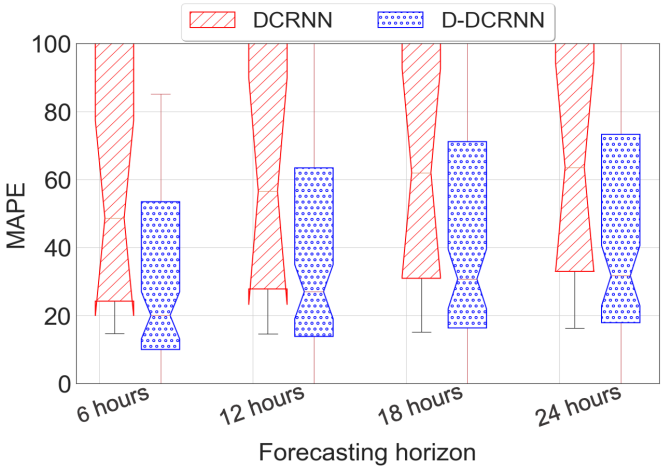


Fig. 9: Distribution of MAPE values obtained by DDCRNN and DCRNN for different forecasting intervals.

We used ARIMA [25] (with order (2,1,2) implemented using the statsmodel [26] Python package); linear regression (LR) [7] (from scikit-learn); random forest [27] (from scikit-learn with default hyperparameters); gradient boosting [28] (from scikit-learn with default hyperparameters); simple recurrent neural network [29] with one hidden layer with 4 neurons (implemented in Keras); long short-term memory (LSTM) [30], stacked LSTM with two hidden layers with 256 neurons per layer (implemented in Keras); gated recurrent unit [31], with one hidden layer with 4 neurons (implemented in Keras); and FireTS [32], [33], a Python package for multivariate time series forecasting with linear regression as the base estimator, autoregression order of 24, exogenous order of 24, and exogenous delay of 0. For each node, we build a node-specific model using each method. Consequently, for a given method we have 96 models. For DDCRNN, we have a single model to forecast the traffic on all 96 nodes. We used 30 hours of input horizon to forecast 24 hours of output horizon. We computed the accuracy values on the test data.

Figure 10 shows the distribution of R^2 obtained by different methods on the test data for different forecasting intervals (1, 3, 6, 9, 12, and 24 hours). The results show that DDCRNN achieves forecasting accuracy values that are significantly higher than those of all the other node-specific models. At the 1st-hour forecast, node-specific model accuracy values are closer to those of DDCRNN; however, they become poor with an increase in the forecasting horizon. Starting from the 3rd-hour forecast, DDCRNN results are significantly better than the node-specific models. From 6th-hour forecast, we observe that only DDCRNN has achieved reasonable accuracy. The superior performance of DDCRNN over node-specific models can be attributed to the former's ability to capture both spatial and temporal patterns using gated recurrent units with diffusion convolution defined on a dynamic graph.

D. Analysis and Characterization

Here we analyse the forecasting accuracy of DDCRNN and provide insights into its strengths and limitations.

Table IV shows 10 nodes where DDCRNN obtained the best and worst R^2 values, respectively. For each node we computed the autocorrelation and partial autocorrelation values with a lag of 10 to analyze to what extent the values in the time series are correlated. Then we computed the mean of the obtained autocorrelation values. Table III shows the mean autocorrelation values for the top three high- and low-accuracy nodes. For the same nodes, the autocorrelation and partial autocorrelation plots are shown in Figure 11.

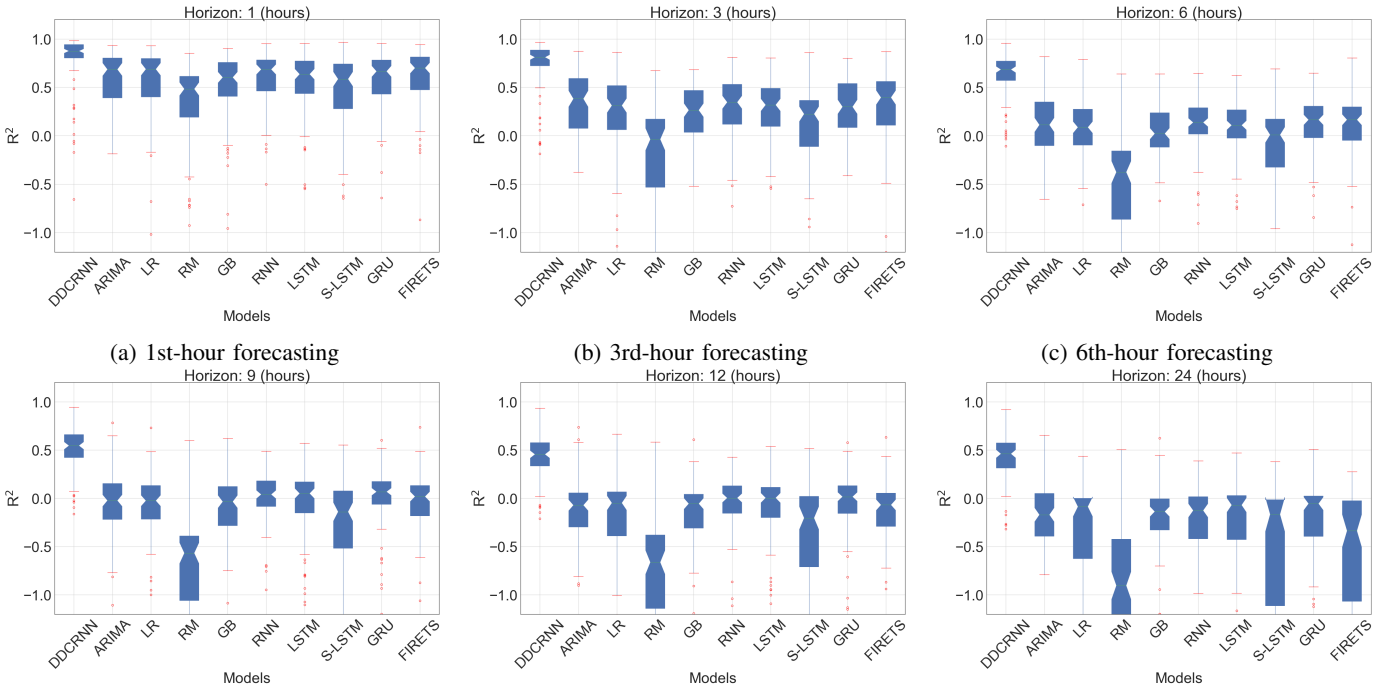
S/n	High Accuracy	Low Accuracy
1	ANL_STAR_in	BNL_NEWY_in
2	ANL_STAR_out	ALBQ_DENV_out
3	SLAC_SUNN_in	CERN-513_WASH_in
4	JGI_SACR_in	NASH_WASH_in
5	FNAL_STAR_out	CERN-513_WASH_out
6	BNL_NEWY_out	ATLA_ORNL_out
7	NERSC_SUNN_out	NASH_WASH_out
8	NERSC_SUNN_in	ATLA_SRS_out
9	SLAC_SUNN_out	ATLA_NASH_in
10	LSVN_SUNN_out	ELPA_HOUS_out

TABLE IV: Nodes with high and low forecasting accuracy.

From the results, we observed that nodes where DDCRNN obtained high accuracy exhibit large autocorrelation values. On the other hand, DDCRNN obtains poor results when there is no strong temporal correlation, where the traffic is random. Upon further analysis, we found that nodes with high forecasting accuracy such as ANL_STAR_in, ANL_STAR_out,

Site	Mean ACF Value
High Accuracy	
ANL_STAR_in	0.783
SLAC_SUNN_in	0.974
JGI_SACR_in	0.842
Low Accuracy	
BNL_NEWY_in	0.238
ALBQ_DENV_out	0.526
CERN-513_WASH_in	0.232

TABLE V: Mean autocorrelation values for the top three nodes with high and low accuracy.



(a) 1st-hour forecasting (b) 3rd-hour forecasting (c) 6th-hour forecasting

(d) 9th-hour forecasting (e) 12-hour forecasting (f) 24th-hour forecasting

Fig. 10: Distribution of R^2 values obtained on 96 nodes by DDCRNN and other node-specific models for different forecasting horizon intervals.

and SLAC_SUNN_in (see Table V) have large experimental facilities. These include Argonne National Laboratory, the Stanford Linear Accelerator, the Joint Genome Institute, and the Linac Coherent Light Source. In these user facilities, the data transfers can happen any time. When a transfer does happen, however, a large volume of experimental data will be moved to data centers for further analysis. These transfers can last several hours and thus exhibit spatial temporal patterns. DDCRNN can learn these patterns to provide high forecasting accuracy. Specifically, our analysis reveals that DDCRNN can model short-term transfers, particularly long connections leading to predictable bulk data transfers especially driven by these large facilities. With respect to low-accuracy nodes, we see a multitude of factors that prevented DDCRNN from reaching high accuracy over a long forecasting horizon. For instance, the CERN facility was shut down most of the year for maintenance, which shows that no transfers or very few patterns are picked up to and from the facility. Additionally, sites such as DENV, BNL, and NASH run performance tests that consist of very small transfers that take up less bandwidth. These transfers are so small that the spatial-temporal patterns are not highly correlated. Therefore, they are not amenable to modeling and learning with DDCRNN. To further test and solidify the technical contribution of our proposed DDCRNN model, we plan to use it on netpredict [34], which is a real-time data framework for testing and deploying multiple machine learning models for network traffic prediction in high-speed R&E networks.

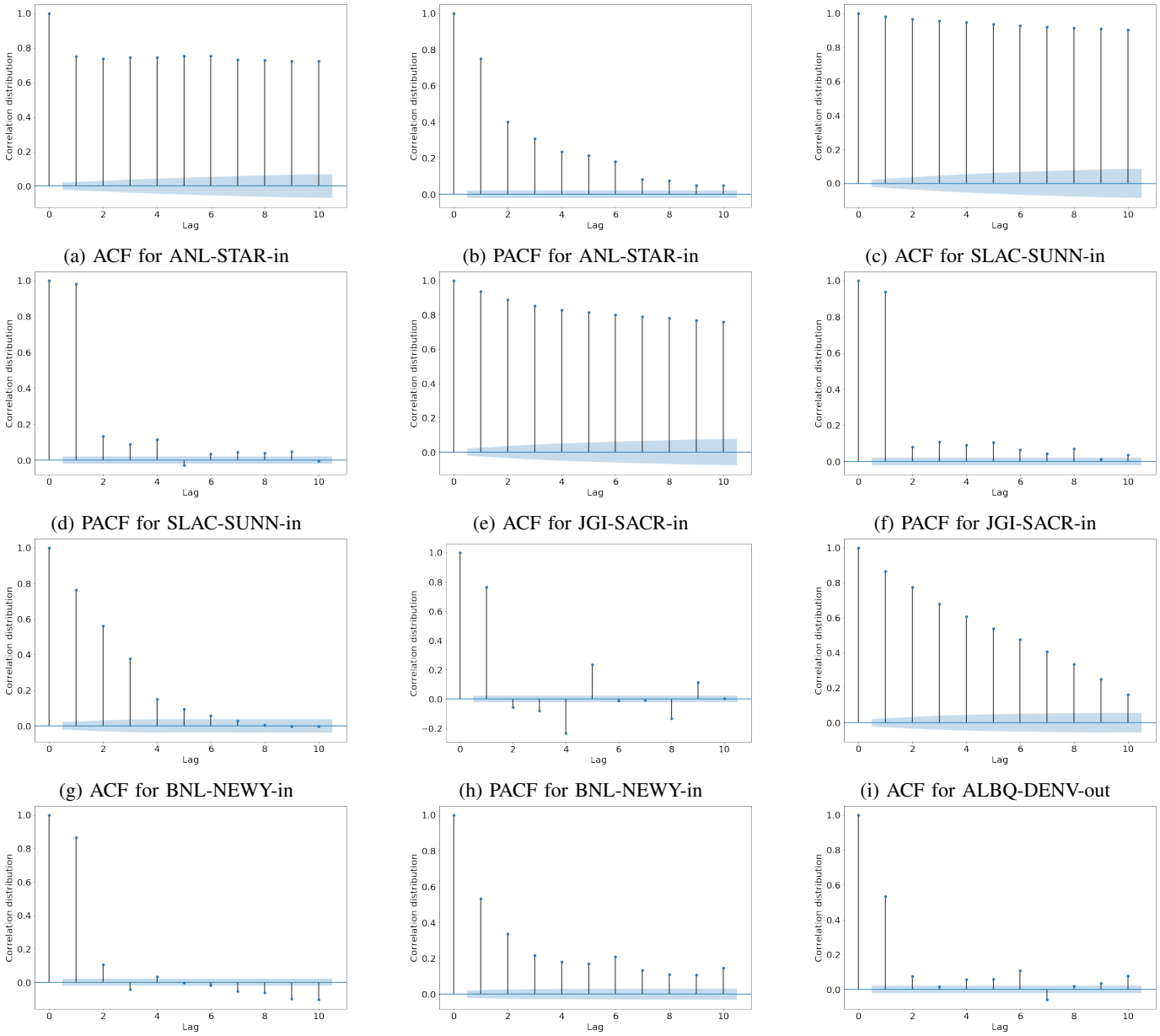
VI. Conclusion and Future Work

We developed DDCRNN, a dynamic diffusion convolution recurrent neural network for forecasting traffic bandwidth in a highly dynamic research wide area network. These traffic traces show dynamic traffic and lack long-term periodic patterns. Our approach is built on a diffusion convolution recurrent neural network that models spatial and temporal patterns using graph diffusion convolution operations within recurrent units. In our approach, the dynamic traffic is explicitly captured by using a state-specific adjacency matrix computed from the current traffic state in the network.

We evaluated our approach on real traffic traces from ESnet, a U.S. research network. Our approach outperformed several other methods. Specifically, the results show that the proposed DDCRNN achieves higher forecasting accuracy than that of the static variant. Moreover, the results show that a single model can be used to achieve higher forecasting accuracy than the site-specific models obtained from linear regression, ARIMA, random forest, gradient boosting, and simple neural network variants that were previously used for traffic forecasting.

Achieving higher prediction accuracy presents several advantages to performing informed routing decisions for potentially new flows that could be impacted with regular R-WAN behavior. Our DDCRNN approach exposes many potentials to allow informed flow and routing allocations for network operations, such as scheduling long-running flow on alternative routes to prevent congestion points for other smaller flows.

In the future, we will couple these decisions with a con-



(a) ACF for ANL-STAR-in (b) PACF for ANL-STAR-in (c) ACF for SLAC-SUNN-in

(d) PACF for SLAC-SUNN-in (e) ACF for JGI-SACR-in (f) PACF for JGI-SACR-in

(g) ACF for BNL-NEWY-in (h) PACF for BNL-NEWY-in (i) ACF for ALBQ-DENV-out

(j) PACF for ALBQ-DENV-out (k) ACF for CERN513-WASH-in (l) PACF for CERN513-WASH-in

Fig. 11: Autocorrelation (ACF) and partial autocorrelation (PACF) plot showing correlation of observations as a function of time lag for different sites.

troller to test how congestion-free routing will impact the average utilization of a network in practice. We will also test and compare with other state-of-the-art baseline models and verify with multiple graphs and several research WAN networks such as Internet2 [35], GEANT [36], and CANARIE [37].

ACKNOWLEDGMENTS

This work was supported by the U.S. Department of Energy, Office of Science Early Career Research Program for 'Large-scale Deep Learning for Intelligent Networks' Contract no FP00006145. This material is based upon work supported by the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357. This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility.

REFERENCES

[1] Cisco, "Trending analysis," *https://bit.ly/2G2zwwD*, 2019.

[2] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9,

no. 1, p. 16, 2018. [Online]. Available: https://doi.org/10.1186/s13174-018-0087-2

[3] k. claffy, "Internet traffic characterization," Ph.D. dissertation, UC San Diego, Jun 1994.

[4] Q. Lu, L. Zhang, S. Sasidharan, W. Wu, P. DeMar, C. Guok, J. Macauley, I. Monga, S. Yu, J. H. Chen, J. Mambretti, J. Kim, S. Noh, X. Yang, T. Lehman, and G. Liu, "Bigdata express: Toward schedulable, predictable, and high-performance data transfer," in *2018 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS)*, 2018, pp. 75–84.

[5] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[6] W. Yoo and A. Sim, "Time-series forecast modeling on high-bandwidth network measurements," *J. Grid Comput.*, vol. 14, no. 3, pp. 463–476, Sep. 2016. [Online]. Available: http://dx.doi.org/10.1007/s10723-016-9368-9

[7] K. Lau and Q. Wu, "Local prediction of non-linear time series using support vector regression," *Pattern recognition*, vol. 41, no. 5, pp. 1539–1547, 2008.

[8] P. Zhao, J. Xia, Y. Dai, and J. He, "Wind speed prediction using support vector regression," in *2010 5th IEEE Conference on Industrial Electronics and Applications*. IEEE, 2010, pp. 882–886.

[9] W. Wang, Y. Bai, C. Yu, Y. Gu, P. Feng, X. Wang, and R. Wang, "A network traffic flow prediction with deep learning approach for large-scale metropolitan area network," *NOMS*, 2018.

[10] B. Zhou, D. He, and Z. Sun, "Traffic modeling and prediction using arima/garch model," in *Modeling and Simulation Tools for Emerging Telecommunication Networks*, 2006, pp. 101–121.

[11] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *International Conference on Learning Representations (ICLR '18)*, 2018.

[12] T. Mallick, P. Balaprakash, E. Rask, and J. Macfarlane, "Graph-partitioning-based diffusion convolution recurrent neural network for large-scale traffic forecasting," *arXiv preprint arXiv:1909.11197*, 2019.

[13] R. Wolski, N. T. Spring, and J. Hayes, "The network weather service: A distributed resource performance forecasting service for metacomputing," *Future Gener. Comput. Syst.*, vol. 15, no. 5-6, pp. 757–768, Oct. 1999. [Online]. Available: http://dx.doi.org/10.1016/S0167-739X(99)00025-4

[14] J. S. Armstrong, "Principles of forecasting: A handbook for researchers and practitioners," *Kluwer Academic Publishers*, 2001.

[15] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, "Cope: Traffic engineering in dynamic networks," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 99–110, Aug. 2006. [Online]. Available: http://doi.acm.org/10.1145/1151659.1159926

[16] M. B. Ntlangu and A. Baghai-Wadji, "Modeling network traffic using time series analysis: A review," in *Int. Conf. Big Data and IoT*, 2017, pp. 209–215. [Online]. Available: http://doi.acm.org/10.1145/3175684.3175725

[17] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[18] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software wan," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013. [Online]. Available: http://doi.acm.org/10.1145/2534169.2486019

[19] A. Pathak, M. Zhang, Y. C. Hu, R. Mahajan, and D. Maltz, "Latency inflation with mpls-based traffic engineering," in *Internet Measurement Conf*, 2011, pp. 463–472.

[20] P. Cortez, M. Rio, M. Rocha, and P. Sousa, "Internet traffic forecasting using neural networks," *IEEE Int. Joint Conf. Neural Network Proceedings*, 2006.

[21] A. Azzouni and G. Pujolle, "Neutm: A neural network-based framework for traffic matrix prediction in sdn," *NOMS*, 2018.

[22] D. Andreoletti, S. Troia, F. Musumeci, S. Giordano, G. Maier, and M. Tornatore, "Network traffic prediction based on diffusion convolutional recurrent neural networks," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019, pp. 246–251.

[23] S. M. Feit, *SNMP: A guide to network management*. McGraw-Hill Professional, 1993.

[24] N. Krishnaswamy, M. Kiran, K. Singh, and B. Mohammed, "Data-driven learning to predict wan network traffic," in *Proceedings of the 3rd International Workshop on Systems and Network Telemetry and Analytics*, ser. SNTA '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 11–18. [Online]. Available: https://doi.org/10.1145/3391812.3396268

[25] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results," *Journal of Transportation Engineering*, vol. 129, no. 6, pp. 664–672, 2003.

[26] S. Seabold and J. Perktold, "statsmodels: Econometric and statistical modeling with python," in *Python in Science Conference*, 2010.

[27] G. Dudek, "Short-term load forecasting using random forests," in *Intelligent Systems' 2014*. Springer, 2015, pp. 821–828.

[28] S. B. Taieb and R. J. Hyndman, "A gradient boosting approach to the kaggle load forecasting competition," *International journal of forecasting*, vol. 30, no. 2, pp. 382–394, 2014.

[29] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, "Time-series extreme event forecasting with neural networks at uber," in *International Conference on Machine Learning*, vol. 34, 2017, pp. 1–5.

[30] S. Selvin, R. Vinayakumar, E. Gopalakrishnan, V. K. Menon, and K. Soman, "Stock price prediction using lstm, rnn and cnn-sliding window model," in *International conference advances in computing, communications and informatics*, 2017, pp. 1643–1647.

[31] R. Fu, Z. Zhang, and L. Li, "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, 2016, pp. 324–328.

[32] J. Xie and Q. Wang, "Benchmark machine learning approaches with classical time series approaches on the blood glucose level prediction challenge." in *KHD@ IJCAI*, 2018, pp. 97–102.

[33] J. Xie, "firets," Aug. 2018. [Online]. Available: https://github.com/jxx123/fireTS

[34] D. Ratner, B. Sumpter, F. Alexander, J. J. Billings, R. Coffee, S. Cousineau, P. Denes, M. Doucet, I. Foster, A. Hexemer *et al.*, "Bes roundtable on producing and managing large scientific data with artificial intelligence and machine learning," DOESC Office of Basic Energy Sciences, Tech. Rep., 2019.

[35] B. Teitelbaum, S. Hares, L. Dunn, R. Neilson, V. Narayan, and F. Reichmeyer, "Internet2 qbone: building a testbed for differentiated services," *IEEE network*, vol. 13, no. 5, pp. 8–16, 1999.

[36] A. M. Koster and M. Kutschka, "Network design under demand uncertainties: A case study on the abilene and geant network data," in *Photonic Networks, 12. ITG Symposium*. VDE, 2011, pp. 1–8.

[37] S. Marcos, "The canadian network for the advancement of research, industry, and education (canarie)," 1992.